ISSN 1744-1986

Technical Report N° 2006/ 30

Communicating Software Requirements: A Comparison of Problem Frames and the UML for Domain Modelling and Requirements Analysis in Commercial Business Projects

Mark Vincent

30 September, 2006

Department of Computing
Faculty of Mathematics, Computing and Technology
The Open University

Walton Hall, Milton Keynes, MK7 6AA
United Kingdom

http://computing.open.ac.uk

Communicating Software Requirements: A Comparison of Problem Frames and the UML for Domain Modelling and Requirements Analysis in Commercial Business Projects

A dissertation submitted in partial fulfilment of the requirements for the Open University's Master of Science Degree in Computing for Commerce and Industry.

Mark Vincent
(PI: W0741924)

13th March 2007

Word Count: 14,992

Preface

I would like to acknowledge all those who contributed to the development of this dissertation and helped me through the research. My sincere thanks go to the following people:

1. My tutor Keith Gordon, for his unfailing support, guidance and encouragement throughout the entire project.

2. Dr Karl Cox and Dr Jon Hall, for reviewing the models used in the research and for the constructive and helpful feedback given.

3. My wife Tracey, for her critical feedback on the finished content as well as her ongoing support, patience and understanding during the project.

4. Finally, I would also like to sincerely thank all those who gave their valuable time to take part in the interviews and undergo the comprehension tests. Out of respect for their confidentiality they have not been named but their input has been invaluable.

Table of Contents

List of Tables

List of Figures

Abstract

The communication about software requirements between software developers and business stakeholders continues to be an area of significant difficulty, and a contributing factor to the high number of projects failing to deliver their intended benefits. Both the Unified Modeling Language (UML) and Problem Frames are proposed as methods to aid this communication, but there appear to be problems in gaining their widespread adoption, especially among the business stakeholders. In the case of Problem Frames, the research to date is still mostly limited to theoretical discussion, with only very few documented examples of industrial use. The UML has a wider commercial case base, but it appears from the literature that there are problems gaining widespread endorsement, especially on the business side.

The research seeks to compare the two methods, to establish which better communicates the problem domain and software requirements to both software practitioners and non specialist business stakeholders. The research focuses upon the notations used and how well they convey their intended meaning, this is achieved through basic comprehension testing of some relatively simple software models using a small test group of participants. The tests were conducted during structured interviews in order to elicit feedback from those who took part.

The findings suggest there are problems with the technical nature of the models, especially on the business side, and this creates a barrier to effective communication. There remains a tendency to use ordinary English text to describe software requirements at a business domain level and it seems this is due to the lack of business acceptance of the existing structured approaches. The way in which software requirements are presented is found to be an important factor in gaining business acceptance. It is proposed that greater consideration should be given to understanding the perspective of non specialist stakeholders, and only by reconciling that perspective with the needs of the software development teams will it be possible to improve the overall communication between business and IT stakeholders.

Chapter 1 Introduction

1.1 Background to the Research

1.1.1 The Difficulties of Commercial Software Development

According to Jackson (2003) the software development process, although now over 50 years old, is "still a curiously immature affair", despite the huge progress made in methods and tools. Indeed, the sheer quantity of literature on software development methods would seem to indicate continuing difficulty for both practitioners and researchers.

On the positive side, software as a product has been a phenomenal success: DeMarco (1993) points to the extraordinary value delivered to business over the years. Even today, investment in software continues at an incredible pace, which indicates that customers must believe in its potential to deliver significant business benefit.

On the negative side, there are serious shortcomings in the software development process, with the computing and business press continuing to report stories of serious project failures. In overall governance terms the picture is somewhat depressing: Thomas (2005) suggests that on average IT projects have a 29 per cent success rate, with an average cost overrun of 59 per cent and schedule delay of 84 per cent.

The failure of IT projects to deliver the expected benefits seems to often stem not so much from the build quality, but from fitness of the finished product for the required purpose. Davis and Hickey (2002) support this view, quoting figures from Standish Group showing that 31% of systems built are never delivered and another 15% satisfied fewer than half the intended customers' needs. They claim this indicates an overall failure in the requirements engineering process.

1.1.2 Communication between Business and IT Stakeholder Groups

Fundamentally, it seems clear that a wider communication problem exists between the business and IT communities in general. Butler (2003) summarises this problem stating:

> "There is still a wide gap between business and IT, with technical staff understanding too little about the requirements of the organisation and the business users finding most technical issues too complex".

In terms of software development, the inherent complexity of the subject matter compounds this problem. Turski (1986) suggests that at the core of the software problem is the need to match properties of the real world, which are difficult to describe, with the constraints of the hardware, which is artificial and man made. Jackson (2003) concludes that the reason software development is so difficult is:

> "…because it combines the mathematic challenge of formal program construction with the scientific and engineering challenges of designing machines to interact with the informal natural human world".

The challenge is therefore to establish a common communication medium where IT and business stakeholder groups can share a conceptual understanding of the problem and agree upon a solution. Within the software literature this is normally deemed to be the primary purpose of domain modelling and requirements analysis. A number of methods have evolved to try to deal with this problem, using various notations and supporting processes. Two of these methods are the focus of this research: The Unified Modeling Language (UML), which has now become a formal standard managed by the Object Management Group; and the Problem Frames approach, developed by Michael Jackson and which is now gaining some momentum within the academic literature.

1.1.3 OOSE and the UML

Object Oriented Software Engineering (OOSE) has gained increasing momentum over recent years. Open University M878/Unit 1 (2000) outlines a key advantage of the approach, which is that "A structural similarity [is maintained] between the world and the software" which "can lead to good traceability from requirements through to code". Another proposed benefit is the unification of modelling constructs and notations into a single standard: the UML.

The UML standard includes support for domain modelling by capturing aspects of the business domain in terms normally recognised by the business users. This is achieved by focusing upon the key concepts existing in the real world, such as orders, products and invoices, and using those as the basis for discussion. The business processes are captured by Use Case models and Activity Diagrams, which describe the ways in which a business domain is used and how it interacts with external entities (Open University M878/Unit 1, 2000).

The UML is already in active use within industry, but it seems there are some issues in gaining business buy-in of the models in their native form. Both Dietzsch (2002) and Arlow et al. (1999) discuss problems with the technical nature of the models and the affects on the business perception of them.

1.1.4 Problem Frames

A method proposed by Jackson (2001): Problem Frames focuses upon the problem environment and the observable changes which should be evident in the 'real world' after the implementation of a software 'machine'. Jackson suggests that existing methodologies tend to focus primarily on the boundary of the required software with the outside world. He asserts that some important details of the problem, while not necessarily occurring at the system boundary, can affect the ability of the software to deliver the required result. He makes the point that ultimately it is the desired observable change in the world that is important to the customer and that all design considerations should be focused towards that end.

Cox et al (2005), reviewing Problem Frames research to date, suggest that some good progress has been made with Problem Frames within academic circles, a significant milestone being the First International Workshop on Advances and Applications of Problem Frames in 2004. However considering it is now over a decade since the idea was introduced, there seem to be very few documented examples of Problem Frames in action within the commercial business world.

1.1.5 Addressing the Communication Problem

Given the complex and abstract nature of software, one could argue that more effort is needed to capture the attention of those without technical expertise. A point raised by Turk and Kirkman (1989), in consideration of the audience of a piece of written text, seems especially relevant to this problem:

> "…if the message is to be communicated effectively, it must be more than just scientifically accurate and grammatically correct. It must also be presented in a manner that commands attention, is quickly informative, and is easily digested by the reader".

Placing this idea in the context of domain modelling and system specification, it is proposed that presentation can help or hinder the process of communication. This appears to be very much the case with domain modelling and system specification. The findings by Dietzsch (2002) suggest that the technical appearance of some models effectively created a barrier to business acceptance. This view was also supported by Arlow et al. (1999), where it was found that models had to be embedded within textual descriptions – an approach they call 'literate modelling' – in order to ensure understanding within the business side. So it seems that many models, while comprehensive and technically accurate, are too abstract for the uninitiated and therefore do not lend themselves easily to widespread acceptance.

1.2 Aims and Objectives of the Proposed Research

1.2.1 Research Aim

Problem Frames and the UML each try to address the communication of problem domains and software requirements in a different way: The Problem Frames approach focuses upon the real world domains and required observable behaviour changes within those domains; the UML builds a definition of the real world in terms of related concepts and behaviours which are ultimately mapped to software artefacts in the final system.

The aim of this research is to establish the relative effectiveness of each approach in communicating the problem domain and software requirements between IT and business people with no previous experience of either method. Given that both approaches originate from within the IT community, there is an implicit assumption that both Problem Frames and the UML will be viewed more positively by the IT stakeholders, but this also needs to be validated.

The research attempts to gain some insight into the ways in which the business and IT stakeholder groups prefer to visualise complex IT problems. It aims to establish which models impart the greatest level of information to each stakeholder through some basic comprehension testing. It also seeks to understand their preferences and how each stakeholder perceives the models in terms of readability and overall fitness for their purpose.

The research question this project attempts to address is:

> *Which method, UML or Problem Frames, better communicates the problem domain and software requirements to both business stakeholders and IT practitioners?*

1.2.2 Research Objectives

In order to answer the question posed, a number of research objectives were defined for this project. These are listed below:

1. Investigate the difficulties existing between the business and IT communities in communicating software requirements and problem characteristics, within typical commercial project environments.

2. Understand the theory of how the Problem Frames and UML approaches support problem domain modelling and requirements analysis.

3. Investigate, through literature case studies, how Problem Frames and the UML have been applied in real commercial projects and any problems identified.

4. Compare the UML and Problem Frames approaches, to assess which tends to convey the greater level of understanding of typical problem domains and software requirements.

5. Investigate the opinions of typical business and IT project participants on the relative strengths and weaknesses of the Problem Frames and UML models, in helping to clarify the problem domain and software requirements.

6. Based upon the results of the research, summarise the findings and establish some recommendations for future development or further research.

This chapter outlined the background to the research, discussed the motivation behind it and defined its aims and objectives. The next chapter considers other related work and discusses how this research is intended to contribute to the current body of knowledge in this area.

Chapter 2 Research Context and Contribution

This chapter discusses other research into the UML and Problem Frames approaches and considers the broader issues of communication between IT and business people. It then places this research in context and explains its intended contribution to the existing literature.

2.1 A Review of the Existing Literature

2.1.1 Problem Frames and the UML – A Brief History

The Problem Frames theory proper was introduced in Jackson (1995) and then refined further in Jackson (2001), with some changes to the elementary Frames. A further discussion on the difficulties of software development and the role of Problem Frames can also be found in Jackson (2003).

The UML was the amalgamation of the work of three authors: James Rumbaugh, Ivar Jackobson and Grady Booch. Their work was eventually combined, together with that of other contributors, into a unified approach and eventually developed into a standard by the Object Management Group (OMG). The key reference for the UML standard at that time was Rumbaugh et al. (1999); this has now been updated (Rumbaugh et al. 2004) to reflect a major revision of the UML standard to UML version 2.

An outline of some of the key research to date for Problem Frames and the UML follows below, together with a discussion about the difficulties of communication between IT and business stakeholder groups.

2.1.2 Research and Discussion on Problem Frames

Cox et al (2005) give a useful overview of Problem Frames research to date. They start by looking at the motivations and guiding principles behind the approach and how it aims to address the problems of software development, focusing on the key differences with other approaches such as the UML. They discuss the techniques involved – including problem decomposition, solution composition and matching Frames to real world problems – and how they have been approached by other researchers. Finally, they consider how researchers have proposed to incorporate Problem Frame based techniques into wider analysis processes such as Problem Domain Oriented Analysis (PDOA), Goal Modelling and process modelling techniques.

Cox and Phalp (2003) propose an approach for deriving Problem Frames directly from process models; they use the Role Activity Diagramming (RAD) process modelling notation to demonstrate how this can be achieved. This idea is then extended further in Cox et al (2004), where a live case study involving an online financial trading system is used to test the concept for real. A number of issues are identified with the matching of Frames to problems and it is suggested that more domain specific Frames will probably be necessary if the approach is to be successful in future. They highlight the difficulty in scoping e-business problems in terms of the depth into the real world which should be analysed; in this case the process models given by the business are used as a starting point.

In Bleistein et al. (2004) and Bleistein et al. (2004a), the focus moves to business strategy as a starting point for development, using a combination of Goal Modelling to structure the requirements and Problem Frames to define the context. A proof of concept case study is carried out using the business model of Seven Eleven Japan to illustrate the process. Jackson's Context Diagram is used to describe the domain context at various levels of abstraction, starting with business strategy, which is also depicted as a Problem Frame. The associated requirements are then linked to the Context Diagrams, but it is suggested that structuring of the requirements is not well covered within the

Jackson approach. A goal hierarchy is therefore proposed to add structure to the requirements and ensure traceability back to the ultimate business goals.

A further discussion on the use of business goals to derive software requirements can be found in Anton and Potts (1998). In this paper they propose the so called Goal Based Requirements Analysis Method (GBRAM). The basic idea is that stakeholders often do not actually know what they want or even agree amongst themselves. However from the stakeholders' wish list, it should be possible to derive the underlying goals of the system and then use these to drive the development process.

Sikkel et al (2000) explore a different perspective, proposing the development of business problem categories based upon the Problem Frames concept. This suggests a higher level view of the problem domain and that recurring problem patterns can exist at that higher level. They propose that typical business software problems can be grouped into one of seven possible categories, which define what is currently understood about the potential solution. At one end of the scale the problem is ill defined and has yet to be properly understood, while in other categories there is a good idea whether, for example, a Commercial off the Shelf (COTS) solution is likely to be required. This allows the potential for different software engineering approaches to be used for the specific type of likely solution.

The idea of higher problem categorisation is also covered in Rapanotti et al. (2004). Here they group combinations of recurring problem and associated solution architecture classes into Frameworks called Architecture Frames or AFrames. AFrames are proposed as a way to guide the difficult process of problem decomposition and solution re-composition. This is achieved through the use of templates to identify which sub-problems and concerns need to be considered. A key benefit of AFrames is that they acknowledge existing architectural styles, together with their inbuilt design expertise, and include them in the consideration of the problem space. This helps to avoid re-inventing the wheel and should, theoretically, lead to increased design re-use.

The combination of UML and Problem Frames is also considered in Choppy and Reggio (2005). They propose a combination of UML constructs and the Problem Frames approach to build domain models, system requirements and design specifications. They suggest a subset of the UML and prescribe specific models, which can be adapted and used at the various stages of domain analysis, requirements specification and design. For each of Jackson's Problem Frames, they define which UML constructs should be used at each of the development stages.

Brier et al. (2005) discuss the concept of so called socio-technical systems, which are the combinations of technology and people required to support typical business process. They propose an extension to the original Jackson Problem Frames, whereby the human behavioural element is modelled as a 'knowledge domain', which interacts with the real world and the machine and is part of the requirement. In doing this, the scope of the problem is increased to include the important human element, allowing the complete business process, and required changes, to be more completely modelled using the Problem Frames notation.

In summary, there appears to an active body of ongoing research and discussion around the Problem Frames approach, although very few examples of live application. This would indicate it has potential but maybe there is still work to be done. In terms of the aims of this research, there appear to have been relatively few attempts to understand the cognitive aspects of the notation and its strength in the communication of complex ideas.

2.1.3 UML Theory and Practice

In addition to the reference manual (Rumbaugh et al. 2004), a plethora of books exist, describing the theory of the UML and its use in domain modelling and software development. The literature included below focuses upon the experiences and discussion resulting from application of the UML in real and experimental situations.

In Arlow et al (1999), the experiences gained from using the UML within a commercial business environment are described using a case study within British Airways (BA). BA has a relatively long history of Object Oriented development and was an early adopter of the UML as a corporate standard. Arlow et al. (1999) outline the challenges involved in achieving business buy-in and understanding of the various models, they propose that embedding the models in supporting text considerably helps the comprehension. They also point to the limitations of visual modelling, leading to the “trivialisation” of important business requirements, which are often hidden away in the notation and giving no sense of the relative importance. In an informal study within BA they discovered that each UML model tended to yield different levels of understanding within each of the stakeholder groups; they categorised these groups as business manager, user, domain expert, analyst, designer and programmer. They found that the preferred model on the business side was the text based Use Case Description, followed by the Use Case Diagrams. Activity Diagrams were not included in the study as they were not supported by their existing Case Tool. There was, however, an expectation that they would have achieved high scores for accessibility and comprehensibility had they been included.

Dietzsch (2002) discusses the experience of using UML within the Swiss Mobilar Insurance Company, who were undergoing a major business process re-engineering exercise. They were looking for a standard notation to support the process engineering phase, the software specification phase and ongoing process improvement. They seemed to have the most difficulty in using the UML within the process engineering phase and suggested there were weaknesses in using it for business process modelling. They specifically cited the technical appearance of the UML as an obstacle for acceptance on the business side. In the end they chose a different method for this part.

Anda et al. (2004) explored the adoption of the UML within ABB, where it was used for the first time on a large safety critical system project. They used interviews with 16 project participants to gain their views of the UML models. Generally they found that the use of UML did contribute to the quality of the final result, although there was significant overhead in the use of the method. They noted in their research some difficulties experienced by the participants in selecting the appropriate models to use in certain cases.

Agarwal and Sinha (2003) tested the ease of use of the UML by surveying a number of MBA and undergraduate students studying Object Oriented Analysis and Design. They found that novice developers generally had a higher initial perception of the UML than those with more experience, suggesting maybe that the theory outperforms the practice. Overall, they found the scores for ease of use of the UML to be disappointing, with state charts performing best, achieving the highest overall rating of 5 out of a possible 7 points.

Otero and Dolado (2002) used 18 final year ICT students to explore the semantic comprehension of dynamic UML models, measuring both the number of correct answers and the time taken to answer. They compared the State Chart, Sequence Diagram and Collaboration Diagram and found a correlation between the model type and the level of comprehension, but this was influenced by the specific characteristics of the domain being described. They propose that certain diagram types suit specific types of domain and suggest further research to understand relationship between diagram semantics and domain.

Finally, Tyndale-Biscoe et al (2002) outline an approach proposed by a small group of companies, as part of the COMBINE project research programme. They suggest that existing UML concepts are aimed more towards the needs of the computer system than the business domain. They propose a way to model business goals and supporting processes using the UML, by defining a set of extension profiles. They also discuss the benefits of the identification of components within the business model, which can then be reflected closely in the system model, thus reinforcing the component based development approach.

It appears from the literature that the UML is widely used within the software development community and has been partially adopted by the business in some cases. However, it seems to be viewed by the business as very much a software developer's tool. It would appear to need further work before it will be fully embraced on the business side as a means of communication with which they are truly comfortable.

2.1.4 The Communication Problem in General

The literature included below outlines some of the problems that exist in the communication between business and IT stakeholder groups.

Martin et al. (2003) explore the deeper communication problems between business and IT people and make the point that information is about generating a "shared, common understanding between people". They suggest a lack of this shared understanding is often at the core of the relationship problems between business and IT stakeholders. They use a case study of a retail bank to research some of the underlying communication issues, using structured interviews to establish how each set of stakeholders views the other. Typically the business view of the IT staff was that they are focused more towards the technology than to the objectives of the business. The IT department was already sensitive to this view and trying to find ways to improve it. The IT department view of the business was that it tends towards ad-hoc behaviour and poor communication, a view which the business was completely unaware of. There was also a view, within the IT department, that the business tended to circumvent project processes such as formal initiation, instead seeing the software development process as something to be challenged by the entrepreneurial spirit.

The conversation between designers and users is explored in Bahn et al. (2002), where two techniques: scenarios and prototypes are compared with traditional methods such as process and data models. The aim is to establish which method results in the greatest amount of feedback from the user. The findings suggested that in the case of prototypes, the users tended to focus on the cosmetics, often forgetting the wider requirements. With scenarios they tended to focus upon the "work situation descriptions" rather than "[software] use scenarios". Neither approach was proposed as a broadly suitable alternative to the use of models in all situations, so it seems there will continue to be a need for models in this area.

A more general discussion about the problems in analysis and design for large projects appears in Curtis et al. (1988), where they explore the human behavioural aspects and the group dynamics in a number of large projects. A lack of application domain knowledge was identified as a constant within these projects and they found that typically a small number of IT specialists, with many years experience within the domain, often had the power and influence over major decisions. The point is that these key people were found to be crucial to project success for their ability to: “map between the behaviour required of the application system and the computational structures that implemented this behaviour”. Interestingly they also highlighted a poor overall perception of documentation within the projects, with teams normally preferring dialogue to clarify issues. They suggested that human and organisational behaviour had a much greater influence on software productivity than tools and methods. The optimistic view would suggest that much has happened to improve this since the paper was written, while pessimists would argue that the fundamental issues are still very much alive and well.

Taking a broader view for a moment, let us consider communication and language in the wider sense. Damasio and Damasio (1992) identify the purpose of language as the transmission of concepts in our minds into the minds of other people, while Seamon and Douglas (1994) define language as a system of gestures, sounds or symbols that embody meaning. They point out that natural languages rely upon clear rules defining the specific sounds and signs and the order in which they must be used. For a natural language, it normally takes many years to acquire sufficient skill to enable free exchange of ideas. Using this parallel, it seems the software community is, in essence, asking the business stakeholders to quickly learn a new language in order to freely exchange ideas, so it is proposed that this learning should be as easy as possible.

The use of pictures or models to convey important information is not limited to the software development community. Symbols are often used in everyday life to communicate important information such as potential hazards, often to diverse audiences with wide ranging literacy and language skills. Wolff and Wogalter (1998) suggest that well designed symbols have the ability to convey large amounts of information very quickly, but if misinterpreted they can have potentially dangerous consequences. The importance of symbols and their role in communicating hazards has even led to the development of national and international standards, giving strict guidelines for evaluating their comprehensibility. Wolff and Wogalter (1998) compare the comprehension testing methods proposed within these standards to establish how the design of the tests themselves can affect the result. An important message from this research is that comprehensibility of even the most basic models should not be taken for granted, and this should be considered carefully in their design.

2.1.5 Summary

Generally speaking, the research to date around Problem Frames has been mainly theoretical, with only a small number of reported case studies. The UML, on the other hand, appears to have already achieved a much wider case base within the commercial business environment. However, from the case studies identified, it seems that very few organisations have adopted the UML as prescribed within the standards. The papers found seem to suggest difficulties in achieving real buy-in from the business side and propose a number of modifications to ‘sweeten the pill’.

While significant effort has been expended in the development of these methods, there remains a reluctance to fully adopt them within software projects. From the experiences outlined above, it seems that the IT and academic communities are still pushing structured approaches more than their peers within the business. This seems at odds with the many articles in the computing and financial press lately on the importance of effective IT governance – which could be viewed as frustration within the business community of the uncontrollable nature of software projects. In other words there seems to be agreement that a problem exists, but no consensus yet on the best way to solve it.

In terms of the broader idea of communication, the notations and semantics used for communicating software requirements are, in essence, other forms of language that must be learned in order to be effective. It seems then that the challenge is to make the language as easy to learn and intuitive as possible. The existence of standards governing the comprehension testing of everyday symbols indicates that even the communication of simple messages is fraught with problems. It follows then that consideration must be given to minimising the chance of misunderstanding within the target audience.

There are clearly some challenges to overcome in improving the communication between the IT and business communities and for now at least it appears to remain a source of great difficulty. Improving the communication is, however, fundamental if significant improvements are to be made in the overall quality of software projects. One can conclude that any method which truly addresses the current communication barrier is very likely to achieve widespread acceptance.

2.2 The Intended Contribution of this Research

Within the literature it seems that communication between business and IT stakeholders is a fundamental issue that has yet to be solved. In commercial projects, the investment must make sense on a business level and deliver sufficient value to the organisation in order to justify the cost of implementation. This suggests that at some level every commercial software project must involve communication between the business and IT stakeholders, and this communication must be effective if the project is to be a success. It seems logical that confirmation of requirements understanding can only be achieved through effective two way dialogue and that this would normally culminate in a representation of the problem and the software requirements that both sides can agree upon. The challenge is to meet the needs of the developers for rigour, precision and completeness of description, while allowing the business stakeholders to communicate using a language with which they feel comfortable.

Both the UML and Problem Frames are proposed as a way forward for software development, but if either approach is to gain widespread acceptance it seems it must capture the hearts and minds of the business as well as the IT community. The common methods for domain modelling have been mainly developed within the IT community and the experiences found in the literature would suggest they are often viewed by the business as difficult to understand. While this seems to be a common and well documented problem, there appear to be relatively few recorded attempts to fully address it.

Focusing on the communication between the two disciplines would seem to be fundamental to making serious improvements in the software development process. Consider once again the Davis and Hickey (2002) view that just over 30% of projects fail even to be delivered. It is accepted that there are often a number of reasons behind project cancellations and not all are down to miscommunication of requirements, but it is certainly cited as a major contributor. In each case, there will have been some investment up to the point of stopping the project, but with little or no benefit delivered. This represents an enormous waste of valuable resource and it would seem that even a small improvement could translate to a significant contribution to the overall bottom line.

Despite the potential cost savings, there seem to be relatively few examples within the literature of attempts to assess domain modelling and requirements analysis techniques from both the business and IT practitioner point of view. Most of the examples of comprehension testing identified seem to focus on students of Information Technology related subjects. There appears to be a need for further insight into the way in which typical business and IT representatives react to the existing models and the difficulties they experience. This research therefore adds to the existing discourse in this area.

Specifically, to the writer's knowledge, there have been no other similar comparisons of the UML and Problem Frames. Therefore, this research attempts to give a new insight into the relative effectiveness of the selected approaches for communication of software requirements. It could also lead to wider discussion on how to fully engage the business stakeholders and thereby lead to improvements in the communication between business and IT.

This chapter discussed the current research into Problem Frames and the UML and highlighted some of the communication problems which exist. It also considered the contribution of this research to the existing body of knowledge. The next chapter describes the research methods to be used and the reasons for selecting them.

Chapter 3 Research Methods

This chapter describes the research methods adopted, discusses how they support the original research objectives and outlines the other research methods considered.

3.1 Research Outline and Aims

The overall aim of the research is to try to establish the relative strengths of the UML and Problem Frames approaches in communicating software related problems and requirements. The selected approach focuses upon those who would typically use the models and aims to understand their reactions to them. Therefore, the core objectives of the primary research are defined as follows:

1. Compare the UML and Problem Frame notations, to assess which tends to convey the greater level of understanding of typical problem domains and software requirements.

2. Investigate the views of typical business and IT project participants on the relative strengths and weaknesses of the Problem Frames and UML models, in helping to clarify the problem domain and software requirements.

The research attempts to establish both objective and subjective results: the objective part through simple comprehension testing and the subjective part by canvassing the opinions of the participants.

The rest of this section describes the research methods used and the reasons for them, in terms of the core objectives defined above.

3.2 Research Approach

Summarising the research objectives, the essence of the research concerns two key factors:

1. The level of semantics conveyed to the reader – the objective part.
2. The perceptions and opinions of the models – the subjective part.

These factors have been considered below in terms of the richness of data, level of resolution and costs associated with the research, as suggested in Open University M801 (2003).

3.2.1 Case Based Methods

Determining the relative semantics conveyed by the models requires a comparative measurement. One approach could be to test the effectiveness of the communication by measuring the delivery of business objectives over a number of projects, maybe using a collective case study approach (Silverman, 2005). While potentially interesting research, many factors influence the effectiveness of communication and the success of projects in general. It would be difficult to isolate the results that are directly attributable to the methods from those of other influencing factors, such as the relative quality of the expert teams involved, the capability of the project teams or the quality of other supporting tools. There is also the problem of selecting projects which are sufficiently similar in all aspects to allow a balanced comparison. The case study approach was therefore ruled out due to the likely cost in terms of the resources required and practical difficulty of carrying out the research.

3.2.2 Semantic Comprehension

Another approach considered, was to use a sample control group of target users of the models and find an appropriate technique to assess the relative levels of comprehension. Wolff and Wogalter (1998) discuss some similar research into the comprehension of day to day symbols, such as warning signs, and point to comprehension testing as the way to achieve this. They confirm that comprehension testing is critical to ensuring that the intended meaning of the symbols is properly conveyed to the target audience. Comprehension testing was deemed appropriate for this research in terms of the level of resolution because it relates precisely to members of a typical project team. It was also achievable within a reasonable cost, in terms of the resources required. Therefore, comprehension testing was selected as the most appropriate method to establish the comparative understanding of the different notations.

3.2.3 Perception and Preference

As discussed in Chapter 1, presentation is very important and if inappropriate it can be a potential barrier to effective communication, so the reactions of the participants towards the models were considered to be of significant value. Therefore, an appropriate method was required to establish how the participants feel about the models. Wilson and Sapsford (2006) list interviews, in their various forms, and questionnaires as the methods most suited to asking questions. Open University M801 (2003) states that a questionnaire has the advantage of lower potential for bias and lower analysis costs, although they need a greater level of preparation in order to be effective; interviews on the other hand tend to produce richer data, but would require more effort in the data analysis.

The chosen approach sought to capture the advantages of both the questionnaire and interview methods, by using structured or “standardised” interviews (Oppenheim, 2001), although “soft-wired” (Rugg and Petre, In press), to allow some free form comment where it was deemed appropriate. Combining the interview with comprehension testing was intended to establish if the perceptions of the participants correlated with the results of the tests, while also giving additional richness to the data by probing some of the underlying factors behind the results.

Balancing the diverging needs for richness of data and analysis cost was a guiding principle during the design of the interview script and the style of the questions.

3.2.4 Sample Selection

One possible approach would be to distribute the comprehension test and questionnaire to a wider anonymous group in order to capture a broader cross section of business domains. However, this option was rejected for the following reasons:

1. Experience shows that the response rates tend to be low where the researcher is unknown to the respondents, unless the interest of the respondent is engaged or the investigation is perceived to be of direct value to the respondent (Wilson and Sapsford, 2006).

2. Ensuring that the sample group, i.e. those actually responding, is truly representative of a cross section of domains would have most likely incurred a much higher resource cost, possibly beyond the scope of a Master's dissertation.

3. As discussed above, it was important to understand not only which method gives a higher score, but also some of the underlying reasons and the nature of any difficulties experienced by the research group. Distribution to a wider anonymous group would fail to capture these difficulties to the same degree, thereby reducing the level of richness of data.

Therefore, a small group of participants, known to the author, was specifically selected to take part in the research, as discussed in Chapter 4.

3.3 Research Preparation

This section describes how the research material was prepared and discusses the choices made, together with the reasons for them.

3.3.1 Developing the Comprehension Test Material

Four simple software problems were selected by the author, based upon a combination of problems found in the literature together with some taken from the author's own experience. All were changed slightly and simplified where appropriate to suit the purposes of the research. The aim was to use problems that would be difficult and varied enough to be challenging, while being based on typical commercial or widely known domains and avoiding highly technical or specialist areas. The objective was to ensure that the problems would have been understandable by all participants if described to them in text form, although sufficiently varied to make it unlikely that any participant had enough prior detailed knowledge of all the domains to materially affect the result.

All problems were introduced at a theoretical point where the business users already have a clear idea about the business processes they want the system to support. In the author's experience this seems to be common for business systems projects. This view is also supported by Cox et al. (2004), who suggest it is especially applicable for e-business systems; they outline a method which uses Role Activity Diagram (RAD) process models as the starting point for deriving Problem Frames. Their approach seemed to give some useful structure, at least to the development of Problem Frames, so a similar idea was adopted for this research.

For each of the four problems, a list of requirements and a RAD model was created, to represent a theoretical discussion with typical business stakeholders. The RAD models and user requirements list should therefore be viewed as the given inputs from the business and were the starting point for developing the UML and Problem Frames models. The following diagrams were then created for each problem:

- UML – Use Case Diagram, Activity Diagram and Class Diagram.
- Problem Frames – Context Diagram and Problem Diagrams.

In both cases, the notations used closely follow the guidance given in Rumbaugh et al. (2004) and Jackson (2001) respectively, avoiding any adaptations found in the research literature. Additional textual notation and context descriptions were also avoided in order to isolate the contribution of the standard notation in conveying meaning to the reader. It is accepted that this could be criticised for a lack of realism, but the aim was to understand how the notations performed in their purest form and therefore how well they aid the communication.

For each requirement, a cross reference was made to the chosen Problem Diagrams and UML model elements. In the case of Problem Frames, this included a brief explanation for the choice of diagram, with the aim being to match as closely as possible the Frames defined in Jackson (2001). For the UML, the specific diagram elements describing each requirement have been individually numbered and cross referenced. A description of each problem, its RAD model, the list of requirements and details of the cross referencing can be found in Appendix A.

It is accepted that some bias may have been unwittingly introduced by the author in the way the models were built. To try to mitigate this, external peer reviews of the models were carried out by two active and prominent researchers within the requirements analysis field, with a view to ensuring that each set of models was a fair representation of the problems described. The goal was to develop equally fair text book representations of each domain within the limitations of each notation.

This chapter discussed the research approach and the reasons for it, in terms of the original objectives. The next chapter describes how the interviews were prepared and conducted and how the data was captured, processed and analysed.

Chapter 4 Data Collection

This chapter describes how the data was collected during the structured interviews and comprehension tests. The criteria used for the selection of participants are discussed, followed by a description of the interview process, and finally the analysis applied to the data to produce the results.

4.1 Research Participants and Background

10 participants were chosen as interviewees, based upon the following criteria:

1. They had no specific familiarity with either of the notations.
2. They had previously been involved in software related projects as a software analyst, project manager or business stakeholder.
3. They were available for a face to face interview with the author.

The interviewees came mostly from within a single organisation, being one of the four major music companies in which the author was working at the time. The primary purpose of the organisation is the creation and distribution of music in its increasingly varied formats. The nature of the music business and the ‘rock and roll’ culture within the company means that there is a tendency towards a less formal approach to projects. No recognised formal methods have been widely adopted at either the project level or within the development process.

The rationale behind using a company of this type was that they are less inclined towards formalism, which poses even greater challenges to any software development approach. In other words, if a method is accepted within this organisational culture, then it should theoretically have a greater chance of adoption within more formal organisations.

Most of the chosen participants were known to the author, which has both advantages and disadvantages. An advantage is that the author was able to select them based upon knowledge of their exposure to software development projects, which it was believed would give a greater likelihood of useful data. It also helped to ensure a good response in terms of securing and executing the interviews. A potential downside of this approach is that the existing relationship and perception of the author by the participant could materially affect the outcomes. In order to guard against this, the interviews were highly standardised (Oppenheim, 2001) and the response options limited where possible; there was minimal intervention from the author during the interview, other than to clarify some points of difficulty by the interviewee.

4.2 Conducting the Interviews

4.2.1 Before the Interview

A short time before each interview, a document was sent to each of the participants, explaining the background to the research and giving them a quick introduction to the modelling notations used; this note has been included in Appendix B.

In the author’s experience, project stakeholders, especially those on the business side, are often presented with new notations with only a very short introduction. They rarely have much time to spend learning the semantics, but they are required to agree specifications that include them. By giving only a brief introduction, it was believed that there would be a much greater reliance upon the meaning conveyed within the diagrams, as opposed to any prior knowledge or training. It was accepted that this could possibly lead to slightly lower comprehension levels for both notations, but the aim was to establish how well the models are perceived and understood by the uninitiated.

4.2.2 During the Interview

The interview was split into three sections:

Section 1 was essentially a spoken questionnaire, which the participant and the author worked through together. The questionnaire, which can be found in Appendix C, focused on the background of the participant and covered the following areas:

- The extent to which they have been involved in software projects and typical roles they would have taken
- Any previous experience, either as a reviewer or developer, of other notations widely used within the software development community, such as Entity Relationship Diagrams
- Any previous experience of the UML or Problem Frames notations

Section 2 focused on comprehension testing, in which each participants was asked to review the four models in turn and list as completely as possible what they understood to be the requirements conveyed by the diagrams. In order to avoid cross contamination, only one type of model for each problem was seen by each participant and the models were equally tested for each of the business problems and for each of the participants. The models used have been included in Appendix D.

Once the participants had finished reviewing each problem they were asked to rate the models for ease of understanding, the responses were captured using a five point numeric Likert style graded preference scale, ranging from 1 (very confusing) to 5 (very easy). There is a good deal of literature relating to Likert scales and the dangers of placing too much reliance upon them. Some research by Guilford et al. (2002) suggests that user perception is likely to differ from person to person, tends to change over time and can be influenced by many factors. However, despite their shortcomings, both Wilson and Sapsford (2006) and Oppenheim (2001) list the Likert procedure as an appropriate and popular method for structuring participant preference responses. To mitigate the possibility of changes in perception due to the passing of time, the ratings were captured immediately after each exercise.

Section 3 was also in the form of a spoken questionnaire and was intended to gain feedback from the participants on how they felt about each of the notations used and any preferences they had developed during the exercises. It was slightly less formal than sections 1 and 2, with more discussion encouraged. The aim was to understand how they felt about diagrammatic models in general and their first impressions of the notations used within the research. It also sought to understand how they have previously felt when reviewing software requirements. On this point the author was trying to establish whether they always feel comfortable with the language used and their understanding of it. The points made by the participants throughout the interview were captured in the additional comments section on the questionnaire form.

4.3 Preliminary Data Analysis

Once the interviews were completed, the results were transcribed from the paper questionnaires into electronic format so that they could be more easily analysed. Using the definitions in Open University M801 (2003) and Sharp et al. (2002), the primary research yielded the following types of data:

Quantitative – The scores from the comprehension tests, the Likert grade preference ratings and the responses from the spoken questionnaire gave a good deal of structured data which could be analysed. For the most part, this data was presented either in tabular form or graphically in order to understand any underlying trends.

Qualitative – The opinions and comments made by the participants were also captured and analysed to establish whether any were common to a number of participants. The common comments and those deemed to be of special relevance were specifically selected for inclusion in the results. Capturing the comments in this way allowed for greater insight into the structured data captured through the comprehension scores and graded preference ratings.

Once captured, the data was analysed to see if any correlations could be found between the different sets of data and the comments made during the interviews. The results of this analysis have been presented in Chapter 5.

This chapter discussed how the data was captured, processed and presented in order to arrive at the final results. The next chapter describes the results found.

Chapter 5 Research Results

This chapter describes the results of the research; it starts with the background of the participants, followed by the comprehension test scores and perceived difficulty ratings, and finishes with the preferences and general comments made.

5.1 Participants' Background and Previous Experience

The first part of the interview focused on the background of the participants and their level of previous experience with software projects. They were asked how many software projects they had previously been directly involved in. All participants had been involved in at least 2 projects and some had considerable experience, having been involved in over 10. The pie chart in Figure 1 shows a summary of the results, with the actual number of respondents (out of the 10 interviewed) shown inside the slice in each case. It should be noted that even for those in the 2 to 5 range, the projects were large in scale and the impact within the organisation was significant.

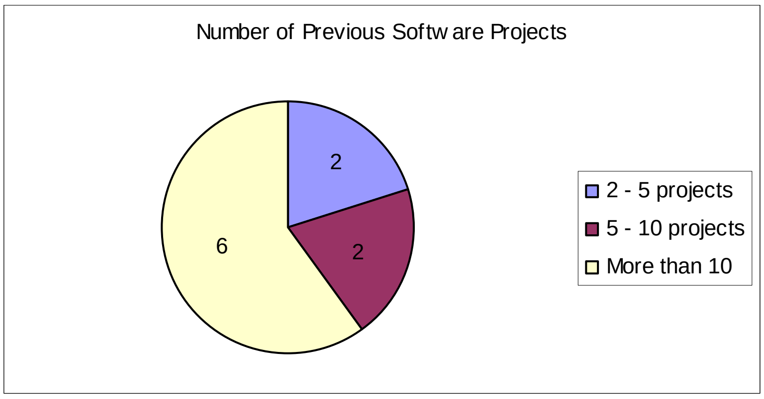


Figure 1 - Pie Chart Showing Number of Previous Software Projects

It was also of interest to understand the type of roles they undertook in those projects. In this research the author has focused on the discussion between business stakeholders and business analysts, so this was the target audience. The pie chart in Figure 2 shows how the roles were spread over the 10 participants.

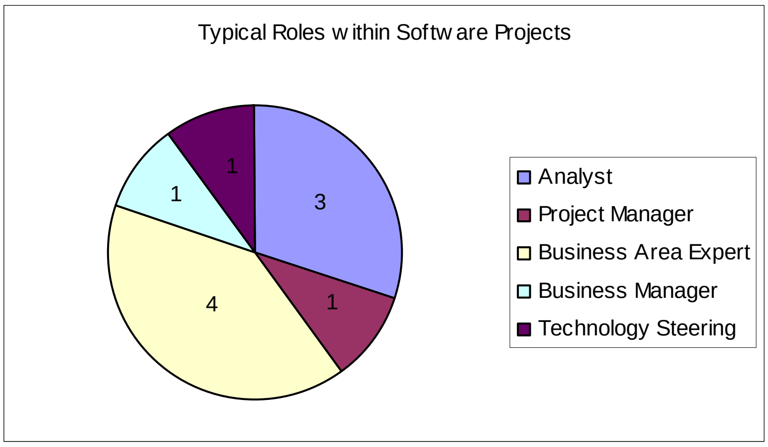


Figure 2 - Pie Chart Showing Typical Roles within Projects

It can be seen from the chart that there were a number of different project roles included in the research; the aim was to try to capture a broad perspective.

5.1.1 Textual versus Diagram Based Specifications

The participants were then asked whether they were used to approving mainly English text or diagram based specifications. The results were that 5 out of the 10 participants were used to mainly text based documents, while the other 5 had experienced a combination of text and diagrams; none were used to working solely with diagram based specifications. It appears from these results that there remains a tendency to use ordinary English text to describe software requirements, at least at a business level. The experience of the sample group was that if diagrams are used they tend to be embedded within text descriptions and used only as supporting information, to help clarify a point.

They were then asked how they prefer to see complex descriptions, whether software or other types of problem. This is a fairly simplistic question and it is accepted that many factors are likely to affect the way it is answered, for example the level of detail at which the participant has been expected to operate within previous projects. The aim was to try to understand whether a perception exists in the minds of the participants of the relative merits of text versus diagrams. The pie chart in Figure 3 shows the spread of responses to this question.

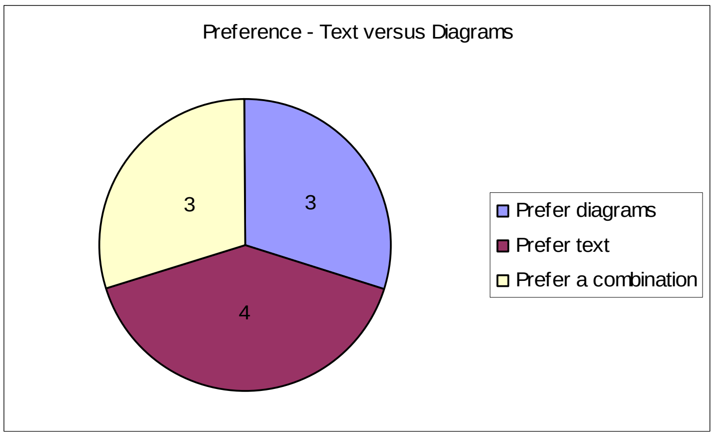


Figure 3 - Pie Chart Showing Relative Preference - Text versus Diagrams

The answers to this question were inconclusive and possibly highlighted a weakness in the question or the way it was asked. However, it was clear from the supporting comments within the meeting, that most participants would expect to see a combination of diagrams and supporting text description, although the relative balance between the two differed according to the individual concerned. One of the respondents, an experienced analyst, suggested that diagrams are good for describing context, but in his experience they struggle when describing the important smaller details. Another analyst also preferred text, but acknowledged that diagrams are useful when discussing among groups as they tend to facilitate easy exchange of ideas. One of the business participants said: “it depends how your mind works, I am unable to relate to diagrams”. It seems from these responses that for domain modelling and requirements analysis plain English text with some supporting – but well annotated – diagrams remains the general preference, especially on the business side. This agrees with Arlow et al. (1999), who found that the UML diagrams were only really accepted within the business when embedded in English text descriptions.

5.1.2 Previous Experience of Software Models

The participants were asked about their experience with software models, both in general and the models used in this research. The aim of this question was to understand the potential for bias due to previous experience of one or other of the notations. The reason for enquiring about some general models is for their similarity with UML models; examples of these are: the Entity Relationship Diagram (ERD), which is similar to the Class Diagram and the Swim-Lanes style process flow diagram, which closely resembles the Activity Diagram. The Activity Diagram also appears to have elements in common with the Role Activity Diagram (RAD), State Charts and maybe to a lesser extent the Data Flow Diagram (DFD), so these were also included.

The participants were carefully selected by the author for having minimal experience with the UML but inevitably it was difficult to exclude exposure completely, especially among the analysts. For those who had some experience of the UML, the experience tended to have been limited to Use Case Diagrams and mostly reviewing only, rather than developing new diagrams. Interestingly, the only participant who had UML Use Case development experience ultimately expressed a preference for the Problem Frames approach and scored higher with the Problem Diagrams than with the UML.

Table 1 below shows the level of previous exposure of each participant to the different types of model. The nature of their previous experience has been denoted as development (D), meaning that they have built these models themselves or review (R) if they have reviewed models built by others.

Participant	DFD	ERD	RAD	Swim-lanes	Use Case
1	D+R	D+R		D+R	R
2	D+R			D+R	
3	R	R	R	R	
4	R				
5	R		R		
6	D+R	D+R		D+R	R
7	D+R	D+R			D+R
8	D+R	R			
9	D+R	D+R		D+R	
10	D+R	D+R	D+R		

Table 1 – Previous Experience of Participants with Different Types of Software Model

There appeared to be a general familiarity in most cases with various forms of software and process model and some exposure to UML use cases. None of the participants knowingly had previous experience of the other UML diagrams or any of the Problem Frames diagrams before doing the exercises.

5.2 Comparing Model Comprehension and Perceived Difficulty

During this part of the interview the participants were shown sets of diagrams, describing in turn each of the four problems. The diagrams were alternated for each participant and the sequence was changed between participants. The actual sequencing of models and participants was as shown below in Table 2, reading from left to right and top to bottom.

Participant	Problem Sequence			
	Problem 1	**Problem 2**	**Problem 3**	**Problem 4**
1	PF	UML	PF	UML
2	UML	PF	UML	PF
3	PF	UML	PF	UML
4	UML	PF	UML	PF
5	PF	UML	PF	UML
6	UML	PF	UML	PF
7	PF	UML	PF	UML
8	UML	PF	UML	PF
9	PF	UML	PF	UML
10	UML	PF	UML	PF

Table 2 - Sequencing of Models by Problem and Participant

The participants were not put under any specific time pressure, but in general the time taken to review each problem tended to be between 5 and 10 minutes. The author observed an interesting difference in the reactions of the participants to the models: the analysts and more technically minded tended to study them very hard first before committing to anything, whereas those from less technical backgrounds tended to read aloud, sometimes making little sense at all initially before they settled on some more coherent answers. One of the business participants was having a particularly difficult time, having worked till midnight the night before and started at 5:00am that morning; this was evident in the level of attention and the resulting scores which were much lower. In the author's experience this is not untypical of the situations sometimes faced by analysts and the interview was therefore still felt to be of value to the research.

5.2.1 Comprehension Scores

During the comprehension exercises, the participants were asked to review each model in turn and describe what they understood to be the software requirements. The answers given and comments made by each participant were then noted by the author; the aim being to establish how many of the original listed requirements had been correctly identified by the participants. The answers given were scored against the criteria shown in Table 3. The full set of participant scores against each requirement can be found in Appendix E.

Score	Meaning
0	The requirement was either not mentioned at all or the understanding of it was completely incorrect.
0.5	The requirement was roughly understood but some important information was missing from the response.
1	The essence of the requirement was clearly seen and understood by the participant.

Table 3 - Scoring Criteria

To get an idea of the relative comprehension between the UML and Problem Frame models, the total scores for each model type were added together for each participant. The graph below, in Figure 4, shows the total comprehension scores for each participant for each model type. Each participant is numbered, to reflect the interview sequence, and has been annotated in the graph with a letter in brackets to indicate whether they were from IT related (I) or business related (B) roles. Higher comprehension scores reflect higher levels of understanding based upon the criteria used.

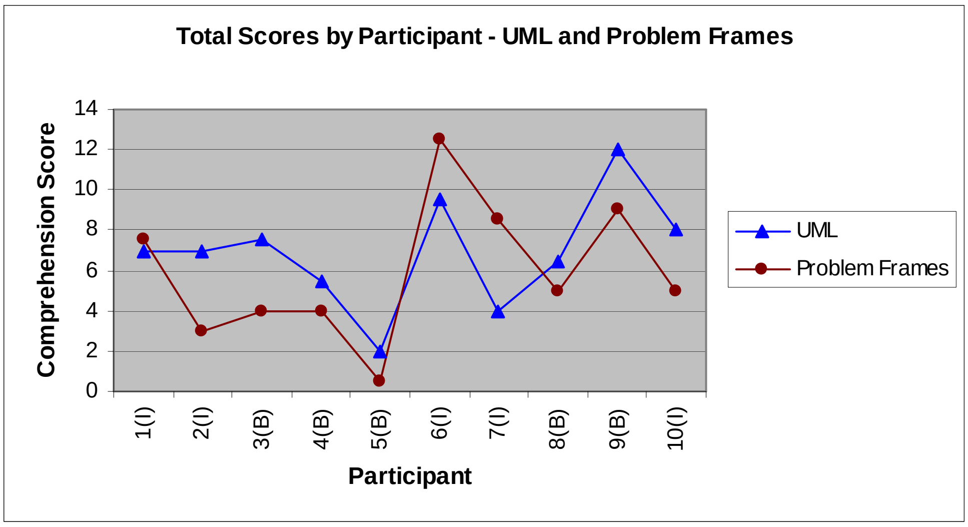


Figure 4 - Relative Comprehension Scores - UML versus Problem Frames

Looking at the chart in Figure 4 it seems that the results are not completely conclusive either way, although the UML is slightly ahead. The scores were also totalled for each notation type across all participants to give an indication of which notation seems to perform best overall. The total possible score for each model type would be 120 if all participants scored full marks. The scores, out of 120 in each case, are shown below.

Notation	**Total Score**
UML	69
Problem Frames	59

Both notations only achieved relatively low scores, which possibly reflects the fact that the participants were not given any idea of context outside the models presented. The reason for taking this approach was to focus the research on the notations and the information they convey, specifically how intuitive they are. It is accepted that this does not represent a typical business situation but that was not the intention, the tests were designed to elicit a more extreme response from the participants and thereby highlight any underlying issues. On the other hand, it should also be noted that these were only very small and simple problems compared to the challenges facing most software engineers, which are typically much larger and more complex in nature.

5.2.2 Difficulty Rating

At the end of each of the four exercises the participants were asked to rate the diagrams, in terms of the level of difficulty in understanding them, using a Likert style scale of 1 (very difficult) to 5 (very easy). Rating scales of this type are not an exact science and have a number of flaws (discussed in section 4.2.2); nevertheless, it was interesting to see whether there was any correlation between the difficulties perceived by the participant and how well they scored in each model. The graphs in Figure 5 show both the scores and the ratings plotted together for each of the problems. On the left axis a higher score indicates greater actual understanding, while the right axis indicates higher perceived understanding (albeit on different scales).

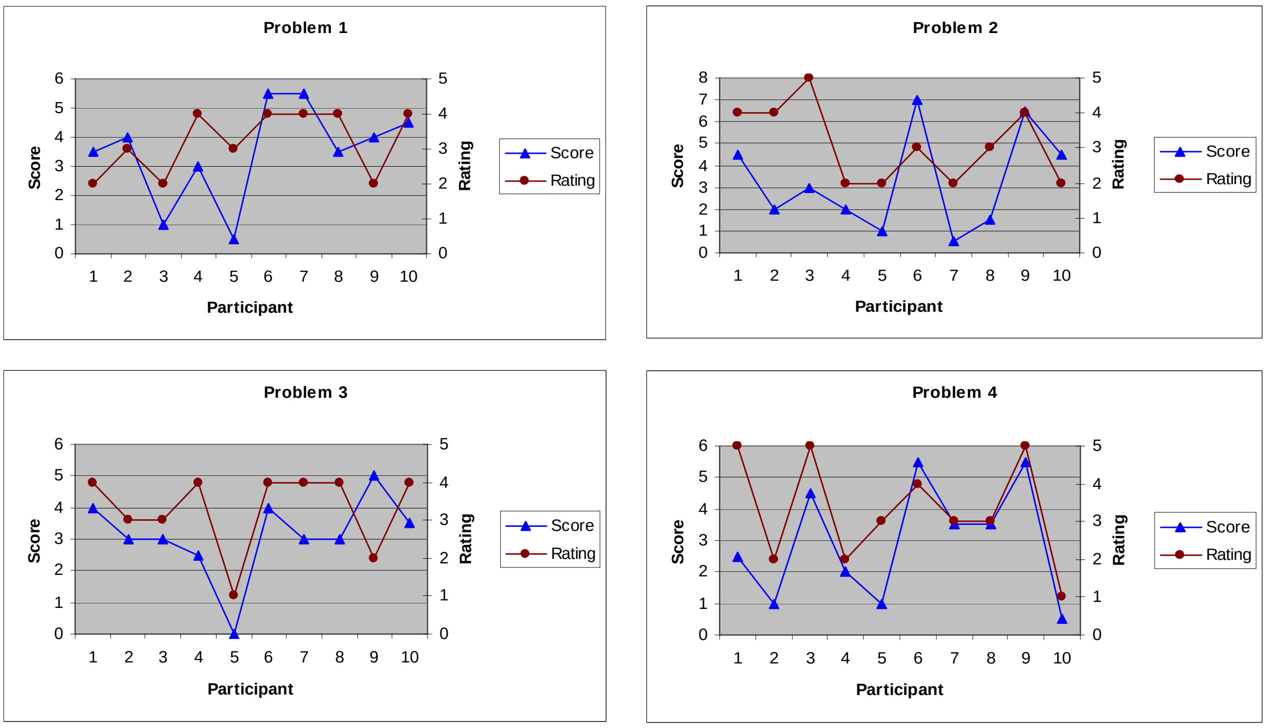


Figure 5 - Comparing Scores and Perceived Difficulty over the Four Problems

There did appear to be a degree of correlation, although this was more marked on some problems than others. While the gap between perception and actual scores varies considerably between participants and problems, there does appear to be a broadly similar profile, in other words higher ratings tend to mean higher scores and vice versa. This would indicate that the Likert style ratings, if not reliable in the absolute, were a reasonable indication of the relative difficulty experienced between the models and between the participants.

To get a comparable number, the total ratings were added for each model type and for each participant. The totals were used to get a view of how the participants perceived the relative level of difficulty of the models. The spread of these ratings over the participants is shown in the graph in Figure 6. As above, the letter in brackets indicates whether they were in IT related (I) or business related (B) roles. Higher rating scores indicate increased ease of understanding as perceived by the participants.

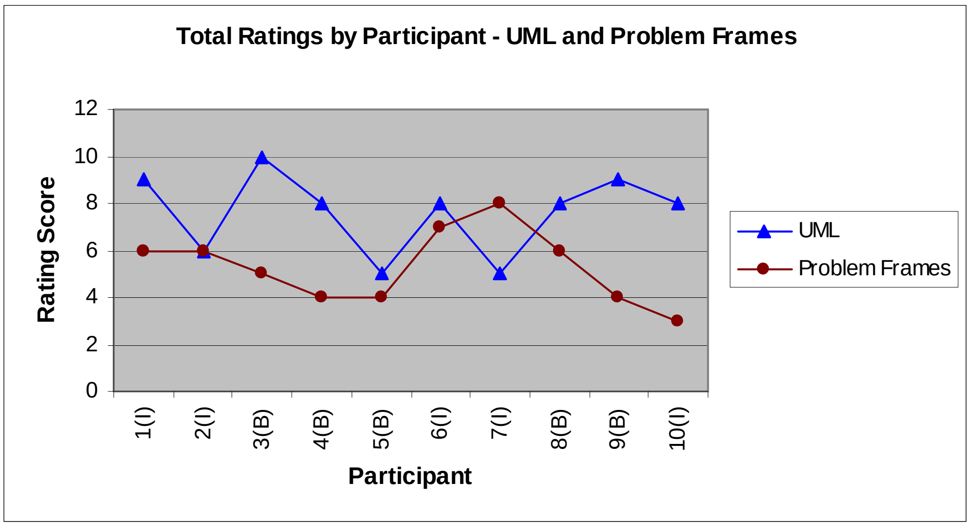


Figure 6 - Comparison of UML and Problem Frames Difficulty Ratings

The rating scores were also totalled for each notation type across all participants, to give an indication of which notation seems to be perceived as the easiest to understand. The total possible score for each model type would have been 100 if all participants gave the highest rating. The rating scores, out of 100 in each case, are shown below.

Notation	**Total Rating**
UML	76
Problem Frames	53

The rating totals indicate a slightly bigger difference between the two notations, with the UML taking a more significant lead.

The totals would indicate a perception from the participants that the Problem Frames notation is more difficult to understand than the UML. However the graph shows that this is by no means universal and the difference varies considerably between participants. Overall there does seem to be a slightly greater differential with the business participants compared to those from an IT background. This was also supported by comments made by a number of IT participants, who stated that they would not feel happy presenting Problem Frames to business people, but could see potential in discussions with software teams. On the business side, without exception, the comments were that Problem Frames are too technical and require considerable effort to understand them.

5.3 Participant Feedback and Opinion

During the final part of the interview, the participants were asked their views on the models they had seen during the exercises and how they compare with other methods of specification they had seen or used in the past.

5.3.1 Perceived Familiarity

First they were asked whether in retrospect any of the models used appeared familiar to them. The answers are shown in the matrix in Table 4 – below – with an X marking those diagrams which, with hindsight, looked familiar.

Participant	Use Case	Class Diagram	Activity Diagram	Problem Context	Problem Diagram
1	X				
2					
3			X		
4	X	X	X		
5					
6	X		X		
7	X				
8					
9			X		
10		X	X		

Table 4 - Models that with Hindsight Looked Familiar

The results show that other than use cases, the most familiar was the Activity Diagram, which was possibly due to its similarity to other well used process modelling notations, as mentioned earlier.

5.3.2 Comparing Ease of Understanding and Preference

The participants were asked which notation they felt was the easiest overall to understand. In this case there was a significant majority who felt that the UML notation was easier, as shown in the pie chart in Figure 7.

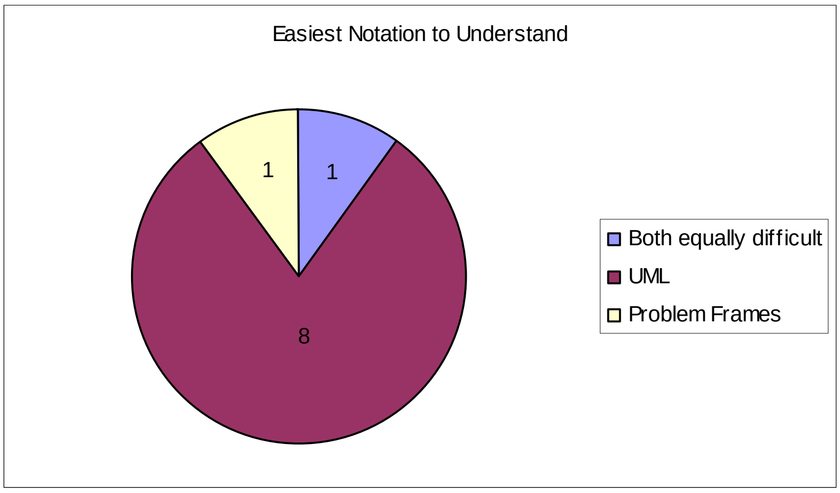


Figure 7 - Pie Chart Showing Easiest Overall Notation to Understand

It should be noted that the comments were more revealing and highlighted the fact that there are also problems with the perceived difficulty of the UML. One participant suggested that both needed a great deal of study and clarification and another said that both were very unappealing in terms of their presentation.

In terms of the specific diagrams, they were asked if there was one they preferred overall, from those they had seen during the exercises. Here the Activity Diagram seemed overwhelmingly to be the most popular, where a preference was indicated, but with one participant preferring the Problem Frames Context Diagram. Figure 8 shows how the responses were divided.

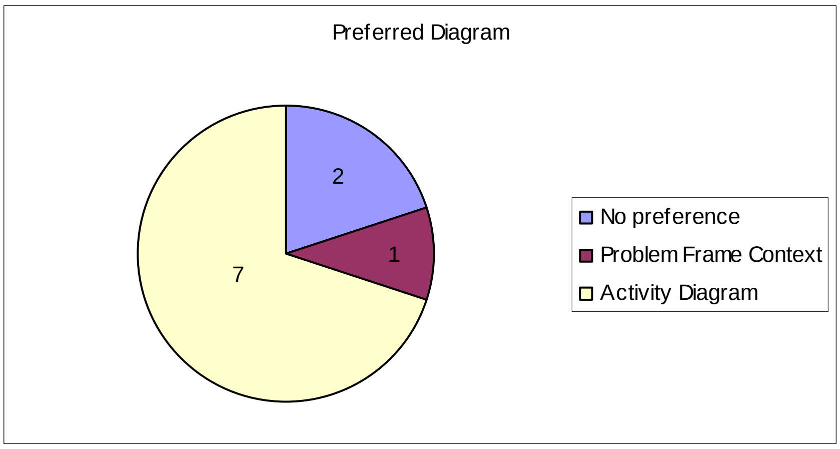


Figure 8 - Pie Chart Showing Preferred Diagram

One regular comment among those who chose the Activity Diagram was that it tends to impart a lot of information in a single diagram. A number of participants also liked the combination of Use Cases and Activity Diagrams, commenting that the two complement each other well and help to give a more complete description of the problem. One rather surprising observation was that although 7 out of 10 of the participants had experience of Entity Relationship Diagrams, none recognised the Class Diagram as being an equivalent view; they tended to either ignore it altogether or in some cases considered it to be an indication of the flow of data between entities.

One IT participant was particularly in favour of the Problem Frames Context Diagram and liked the way it focuses on the essence of the problem, while ignoring the technical concerns. Other IT participants expressed interest in the Problem Frames Context Diagram and the overall concept, but were concerned about the proliferation of Problem Diagrams and the fragmentation of the problem.

The participants were then asked how the models used in the exercises compared to others they had seen or used in the past. The distribution of responses is shown in the pie chart in Figure 9.

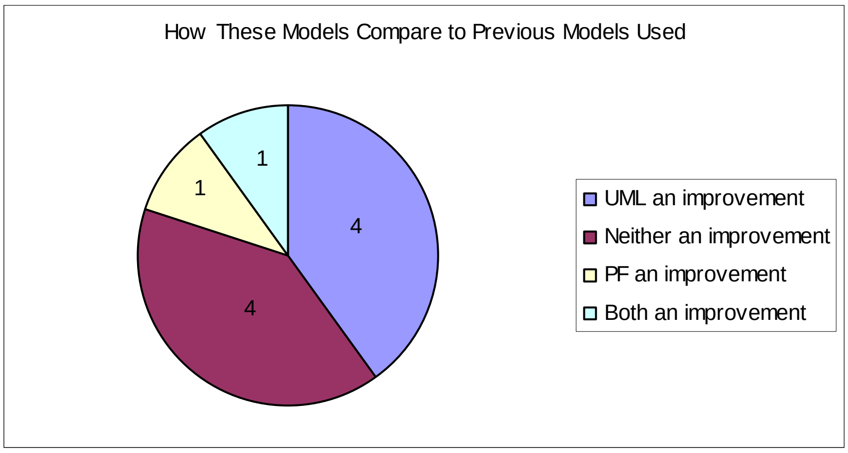


Figure 9 - Pie Chart Showing How Participants Rated the Models Compared to Others

This question is somewhat unfair on both notations because the models were intentionally unrealistic, in that there was no supporting text and minimal annotation of the diagrams. However 5 out of the 10 respondents considered the UML to be a step forward, compared to specifications they had seen previously, while only 2 out of 10 thought the Problem Frames notation was an improvement. Again, both respondents who considered the Problem Frames notation to be an improvement were from IT related backgrounds.

5.3.3 The Language of Software Requirements

Finally, the author asked the respondents to discuss how they have previously felt about the language and documentation used to describe software requirements. They were asked to consider, honestly, the last specification they reviewed and indicate how they felt when reading it, by selecting from the following possible responses:

1. I felt very comfortable that the requirements were clearly and completely documented, both in summary and in detail.
2. I felt ok in general, but didn't fully understand all the finer detail, that's up to the developers.
3. I actually understood very little, but I'll just hold the project team accountable if it is wrong.

Generally, most participants stated that they would only accept a specification once they had fully understood it. A small number admitted they had not always understood all the details and suggested that this would depend upon management style and the level of trust in the team. Those who said they would always fully understand before approving a specification admitted they had normally invested a significant amount of time understanding it. One participant stated that he had insisted on significant rewriting of some parts of the specification into a format with which he felt comfortable.

5.4 Summary

This chapter presented the results of the research; it included the scores for the comprehension testing exercises and Likert style ratings, together with some discussion of the main comments made by the participants during the interviews. The next chapter reviews these results in the light of other similar research and suggests the conclusions which may be drawn from them.

Chapter 6 Analysis of the Results

This chapter discusses the results of the research and suggests conclusions that may be drawn.

6.1 Comparing the Relative Comprehension

The overall comprehension scores for both notations were relatively low, with the participants scoring a total of 69 for the UML and 59 for Problem Frames out of a possible 120. The exercise was designed to test the ease of understanding when viewed without any prior knowledge of either the domain or the notation used. It is accepted that this does not reflect a wholly realistic situation; nevertheless, the author believes the low scores are a cause for concern, especially as the problems described were all very small and simple compared to those encountered within most commercial software projects. In the author's experience, project members (especially on the business side) are often required to read and understand models for which they have no prior experience, so it is proposed that this should be considered in the design of the notation. These results, if truly reflecting the levels of comprehension achieved within typical projects, would seem to indicate a very high potential for error in the requirements engineering activity.

The results varied considerably between participants, suggesting that there is not a single approach that will necessarily suit everyone. It seemed to be reasonably clear though, both in the comprehension tests and the preferences, that the business users where generally less comfortable with the Problem Frames diagrams than those with an IT background. In all 3 cases where the Problem Frames diagrams scored higher than the UML, the participants were from IT related roles. Of the remaining two IT participants: one is a very much a business focused analyst, originally from a less technical background and the other a very senior technology manager, used to working at much higher levels of abstraction. It would appear from these results that there is a greater affinity for Problem Frames from within the IT community. This seems surprising, as the focus of the

Problem Frames approach on structuring the problem domain would suggest that it should, in theory, be more amenable to the business stakeholders.

When plotted in the graph in Figure 4, the total comprehension scores for Problem Frames and the UML show some correlation between the two methods such that a higher score in one tends to lead to a higher score in the other. This would suggest that some people are simply more comfortable than others with models. In fact this point was made by one of the participants, who stated that in general he feels more comfortable with ordinary English text than with diagrams.

One particularly concerning observation was that in a number of cases participants felt they had understood the requirement, but the answer they gave was quite wrong. The implication of such misinterpretations in real live projects could potentially be very costly or even dangerous. This part of the research paralleled the work of Wolff and Wogalter (1998), who stress the importance of ensuring the correct interpretation of safely signs and highlight how easily even the simplest symbols can be misunderstood. It is the author’s belief that this should be given serious consideration in the development of notations used to communicate complex ideas. It is proposed that measures should be taken to ensure, as far as possible, that the intended semantics are properly understood by the recipient and take into account their background and level of technical knowledge.

6.2 Participant Perceptions and Comments

The overall perceptions of the participants showed a slightly greater differential between the two methods than the comprehension scores. The UML scored higher, with 76 out of a possible 100 points, versus 53 for the Problem Frames notation, but again the difference varied considerably between participants. This time only one participant (from an IT background) gave the Problem Frames diagrams a higher rating.

The comments given by the participants were that the Problem Diagrams were too fragmented and did not give a good sense of the whole problem; of the Problem Frames diagrams the Context Diagram was the one preferred by most participants. A similar point, about the difficulty of structuring the requirements within the Problem Frames method, is made in Bleistein et al. (2004) and Bleistein et al. (2004a), where it is proposed to use the Context Diagram in combination with Goal Models rather than Problem Diagrams. Another notable area of difficulty with Problem Frames for most participants was the notation used for the interface descriptions; even though they had all been briefed on the notation in writing as part of their preparation, it was still necessary to explain the meaning during the interview in every case.

The preferred diagram by a large margin was the UML Activity Diagram and the reason given was that it gives a good sense of the overall requirements in one single diagram. It was also interesting to note that this opinion was shared across both business and IT participants. Therefore, it appears that both groups prefer to focus on the dynamic aspects of a problem, such as the flows of information or passing of responsibility between elements of the problem. The Use Case diagram was also mentioned as very useful and many suggested that it works well when viewed alongside the Activity Diagram. It seems likely that the popularity of the Activity Diagram is also linked to the fact that it was also the model that recorded the highest level of familiarity; probably due to its similarity to other widely used process modelling notations. It could be that a sense of perceived familiarity helped the participants to feel more comfortable with the UML, even though they did not have direct experience of it. This could have given the UML an advantage, with regard to the perception of the models, and may have even contributed to the higher comprehension scores. A conclusion that could be drawn from this is that any notation which is to be widely accepted should adopt, as far as possible, existing well recognised notational constructs.

The least popular of the UML notations was the Class Diagram, which tended to be either ignored or interpreted as another data flow diagram. This was surprising considering that most participants claimed to have some experience of Entity Relationship Diagrams. It seems likely that either they misunderstood what Entity Relationship Diagrams are when asked the question or that possibly their involvement has been very limited or took place some time ago. It is proposed that for most people, especially on the business side, the dynamic aspects of a problem seem to be conceptually easier to grasp than the static perspective, which tends to be more abstract.

There are only very few examples of Problem Frames in live commercial use, so it is difficult to be certain that the business view of them identified in this research can be generalised. Cox and Phalp (2003) seem to indicate a positive response from the business users to the method, although there is no indication of how they felt about the notation. Indeed, some participants in this research expressed an interest in the Problem Frames concept, but seemed confused and alienated by the notation, which was seen as technical and difficult to read. Considering the view of Turk and Kirkman (1989), that presentation is important when communicating a message; it may be an indication as to why there has been only a limited endorsement of the method, especially on the business side.

Presentation seemed to be a key consideration, especially with the business participants: one participant with considerable experience of software projects was unimpressed by all the models shown, stating that they do not capture the imagination; another considered some of the models to be "a clumsy way of explaining something". Considering the responses given and comments made by the participants, it would seem that plain English text, embedded with some appropriate and clearly annotated diagrams, remains the communication medium of choice for describing software problems and requirements.

There are well recognised problems with using plain English text in achieving the required precision of description for software development. It is, therefore, important to develop an alternative communication language which meets the needs of both sides. A point specifically raised by one participant on the business side was that the language used by the software community is designed to exclude those who must ultimately accept the final product. This problem is also discussed in

Martin et al. (2003), where it is suggested that a healthy business communication culture relies upon a system of shared meaning, achieved through the use of a common language that suits the needs of both parties. It can be concluded from this research that there is still work to be done if such a language is to become reality.

6.3 Summary

The research found that the language and models used by the software development community are not yet sufficiently well suited to the needs of the business stakeholders. There remains a preference at the business level for descriptions in a language which is easy for them to understand, without the extensive use of technical terms or specialist concepts. Martin et al. (2003) suggest that a lack of accountability on the business side is also a contributing factor and that both sides must work at building “a shared information culture”.

The problems described by the models were relatively small and simple compared to those normally found in the commercial environment. Taking this into account, it could be concluded that the results were relatively poor in terms of both the comprehension levels and the perceived difficulty. A particular concern was the fact that in some cases the diagram elements were completely misinterpreted and understood to mean something quite different. While the limitations of the exercise are accepted, this is a potential danger for projects where both sides believe there is a common understanding, when in fact each party has a materially different interpretation.

The Problem Frames approach, while potentially a very powerful concept, seemed to fare relatively poorly in these exercises compared to the UML. This appeared to be because the models are too technical in appearance and maybe too abstract to be easily digested by those who are not extensively briefed in the method. Indeed the author’s own experience while researching the approach was that the concept, while very powerful once understood, is not easy to grasp, even for those with extensive software development experience. It is proposed that this may have been one of the main barriers to a widespread adoption both within and outside the software community.

This chapter discussed the results of the research and considered their implications in terms of the communication of software requirements. The next chapter summarises the research and suggests potential areas of further work.

Chapter 7 Conclusions

This chapter summarises the research in terms of the original project objectives, it considers the research methods used and comments on how effective they were. Finally, some suggestions for potential future research are proposed.

7.1 Research Review in Terms of the Objectives

The focus of this research was upon the communication of software requirements between IT and business stakeholders. Chapter 1 identified this as a very difficult area, requiring the reconciliation of the needs of business stakeholders on one hand, for richness and simplicity of description, while also developing specifications that are rigourous, complete and detailed enough to build a software solution.

The research sought to compare two prominent software development notations: Problem Frames and the UML, to try to establish which is more effective at conveying the problem domain and software requirements. In answering this question it also attempted to gain some insight into the ways in which the business and IT stakeholder groups prefer to visualise complex IT problems.

Chapters 1 and 2 outlined the communication difficulties existing between the business and IT communities. A number of problems were identified in the literature, both in terms of the general communication and specifically within the requirements analysis activity; it seems that a common discussion language for software related problems is urgently needed.

Chapter 2 reviews the current research into Problem Frames and the UML. It finds that the Problem Frames research is still mostly theoretical, while the UML has a number of examples of industrial use. In terms of business adoption, the UML approach seems to have had some limited success, a notable example being British Airways, for which it has been a standard for many years. However, this is not widespread and in each case it required adaptation to suit the business' needs. The Problem Frames approach has only very few live examples, but from the case studies identified it appears to have received a reasonably positive response from the business users.

Chapters 3 and 4 described the primary research. The research focused on a comparison of the graphic notations used with the UML and Problem Frames methods and how much information they convey when presented cold and without supporting context. Within this scope, the two methods were compared by asking a number of participants to review some simple models from each notation and list the requirements being described in each case. They were also asked to rate the models in terms of the level of difficulty and comment on their findings. The comprehension tests and graded perceived difficulty responses were embedded within a structured interview format in order to capture as much of the participant feedback as possible. The selected participants were from a number of roles and split equally between IT and business related backgrounds.

Chapters 5 and 6 described the results of the research and found that the UML scored slightly higher in both the comprehension and perceived difficulty ratings, although the results varied considerably between the different participants and problems. In general the business participants seemed to find the Problem Frames diagrams more difficult than those from an IT background; this was evident from both the scores and the feedback.

The scores for both notations were relatively low, which is concerning given the simplicity of the problems described. While it is accepted that there is far more to the development of a shared understanding than simply the notation, it is proposed that if they are to add real value and give a sound basis for specification signoff, then they must be clear, unambiguous and communicate their meaning easily. Most importantly, they must convey the correct meaning and leave little chance of misunderstanding. The key to this appears to be in considering very carefully the needs of the non specialist stakeholders, when discussing complex software problems, and to reconcile them with the painstaking detail and structured description required by the development teams. The difficulty of reconciling these very different needs would appear to be at the core of the communication problems between the two sides.

In the case of the Problem Frames method, it is clear from the literature that it is about far more than just the notation: it is about considering software problems in a different way. There appears to be considerable value in the approach, but the results suggest that there are some difficulties with the notation, which does not appear to lend itself to easy comprehension by non specialists.

The UML scored slightly higher in the tests and has achieved some penetration into the commercial business environment. However, it seems that it is seen as very much a software developers language which the business side has some difficulty in learning. Tyndale-Biscoe et al (2002) also highlight a similar issue and point to the work of the COMBINE initiative, which proposes extensions to the UML to enable richer business level description.

The findings of this research also concur with those of Dietzsch (2002) who, when reviewing the use of the UML as part of a process re-engineering exercise, found the technical appearance of the models to be “an obstacle for acceptance on the business side”.

7.2 A Review of the Research Methodology

The selected method of comparison: comprehension testing, has given some insight into the contribution the models themselves would make to the requirements engineering process. It is proposed in this research as a useful way of identifying whether the models are likely to help or hinder the communication of requirements. It parallels similar research into safety warning signs and the importance of ensuring that the received meaning is as intended. However, it is accepted that comparison of the methods in this way only considers one dimension of the problem, whereas communication is a much wider process, of which the models form only part. Models of this type are normally the result of many hours’ discussion and feedback between interested parties and in the author's experience it is this discussion, if managed correctly, that ultimately delivers a shared understanding; a point also made by Curtis et al. (1988).

A concern with this type of research is that the test material – in this case the models – may have unwittingly been biased towards one or other of the methods. Although independent reviews were carried out on the models by well known and respected researchers, it is difficult to be sure that both notations had an equally fair representation. In this case, particular difficulty was experienced with the structuring of the Problem Diagrams and it could always be argued that they may have been structured differently to improve the results. A balance was sought to create an equally fair representation of each notation, within the bounds of the standard literature, while not invalidating the comprehension tests by spelling out the requirements in plain text descriptions within the models. Overall, it is believed that even taking into account these concerns, the results gave an indication of the likely challenges facing the two approaches, at least from the business side.

Gathering the feedback of the participants in the form of graded responses would probably have had limited value if taken in isolation, taking into account the problems with Likert style grading systems and inherent unreliability of human perception. However, when combined with the comprehension tests and the comments from the interviewees, they seemed to give an additional and very useful insight into the perceptions of the participants and the difficulties they experienced. To some extent the graded perceived difficulty results appeared to be a magnification of the findings of the comprehension tests, for example highlighting the differences between the business and IT participants more obviously. This would appear to reinforce the importance of presentation in the communication of a message.

7.3 Further work

Both Problem Frames and the UML can play an important role in the software development process, but in their current form it would appear that they are less than ideal for communicating to non specialists. Reconciliation of the different requirements – of the business on one hand and the software development teams on the other – would seem to be a fundamental step in the development of a language that truly resolves the current IT and business communication difficulties. Future research should therefore first focus upon clarifying those needs, especially from the point of view of the non specialists. It is proposed that such research should be placed in the wider context of cognitive and communication theory and would seek to understand more clearly how different groups of project stakeholders visualise and communicate complex ideas.

The scope of this research was limited to a single company, which may have possessed certain characteristics not widely shared in the commercial world. Therefore, it would be useful to understand if the findings represent a general pattern across a wider perspective; for example by carrying out similar research across a wider cross section of participants from a number of different commercial and non commercial sectors, to establish if the results can be replicated.

Intuitively, one could conclude that the Problem Frames approach should be more amenable to the business stakeholders, given its focus on the problem space and desired behaviour changes in that space. In commercial business projects, it is the combination of people and systems that deliver business value through business processes. Brier et al. (2005) raise some interesting ideas in this area and their research may yield an opportunity to engage the business stakeholders more closely, by discussing problems and solutions at a business process level. Nevertheless, the findings here would suggest that adaptations are needed to the presentation of the Problem Frames concept and the supporting notational constructs, in order to make the approach more easily digestible by non specialists. Making the Problem Frames concept more easily accessible to a wider audience would, therefore, seem to be a very worthwhile area for further research.

It appears from this research that much of the effort in creating software development methods has been focused inwardly towards the needs of the software development teams, with less consideration for the needs of the non specialists. There seems to have been little success in building a software and problem domain description language that is fully adopted by the business, while also meeting the needs of software engineers. Ultimately, the effectiveness of any development approach will be judged by those who need to use it and the true measure of success must surely lie in how widely it is adopted.

References

Agarwal, R. and Sinha, A.P. (2003) 'Object-oriented modeling with UML: a study of developers' perceptions', *Communications of the ACM,* vol. 46, no. 9, pp. 248-256.

Anda, B. and Hansen, K. (2005) *An Empirical Study of Use Case Modelling in a Large Development Project*, [online], http://webseminar3.unix160.cn4e.com/xprogrammer/source/a_case_study_on_use_case_modeling.pdf. [Accessed 12th May 2006].

Anton, A.I. and Potts, C. (1998) 'The use of goals to surface requirements for evolving systems' *Proceedings of the 20th international conference on Software engineering, April 1998,* IEEE Computer Society Washington, DC, USA, pp.157-166.

Arlow, J., Emmerich, W. and Quinn, J. (1999) 'Literate Modelling — Capturing Business Knowledge with the UML' *The Unified Modeling Language. «UML»'98: Beyond the Notation: First International Workshop June 3-4 1998, Mulhouse, France,* pp.189-199.

Bahn, D., Naumann, J.D. and Curley, S. (2002) 'Conversation about Software Requirements with Prototypes and Scenarios' in Halevy, A. and Gal, A. (eds.) *Lecture Notes In Computer Science archive. Proceedings of the 5th International Workshop on Next Generation Information Technologies and Systems June 24-25 2002,* London, pp.147-157.

Bleistein, S.J., Cox, K. and Verner, J. (2004) 'Problem Frames Approach for e-Business Systems' in Cox, K., Hall, J. and Rapanotti, L. (eds.) *International Workshop on Advances and Applications of Problem Frames (IWAAPF) May 2004*, pp.7-8.

Bleistein, S.J., Cox, K. and Verner, J. (2004a) 'Requirements engineering for e-business systems: integrating Jackson problem diagrams with goal modeling and BPM' *Software Engineering Conference, 2004 11th Asia-Pacific,* pp.410-417.

Brier, J., Rapanotti, L. and Hall, J.G. (2005) *Capturing Change in Socio-technical System with Problem Frames,* 2005/01, Milton Keynes, UK, The Open University.

Butler Group (2003) *Technology Evaluation and Comparison Report,* Butler Direct Limited.

Choppy, C. and Reggio, G. (2005) 'A UML-Based Approach for Problem Frame Oriented Software Development', *Information and Software Technology* [online], http://www.disi.unige.it/person/ReggioG/ChoppyReggio05a.pdf. [Accessed 12th January 2007].

Cox, K., Hall, J.G. and Rapanotti, L. (2005) 'A roadmap of problem frames research', *Information and Software Technology,* vol. 47, no. 14, pp. 891-902.

Cox, K. and Phalp, K. (2003) 'From Process Model to Problem Frame–A Position Paper' in Achour, C.S.B., Regnell, B. and Kamsties, E. (eds.) *9th International Workshop on Requirements Engineering–Foundations for Software Quality (REFSQ'03),* pp.93-96.

Cox, K., Phalp, K.T., Bleistein, S.J. and Verner, J.M. (2004) 'Deriving Requirements from Process Models via the Problem Frames Approach', *Information and Software Technology,* vol. 47, no. 5, pp. 319-337.

Curtis, B., Krasner, H. and Iscoe, N. (1988) 'A field study of the software design process for large systems', *Communications of the ACM,* vol. 31, no. 11, pp. 1268-1287.

Damasio and Damasio (1992) quoted in Seamon, J.G. and Kenrick, D.T. (1994) *Psychology,* (2nd edn), Englewood Cliffs, New Jersey, Prentice Hall inc.

Davis, A.M. and Hickey, A.M. (2002) 'Requirements Researchers: Do We Practice What We Preach?', *Requirements Engineering,* vol. 7, no. 2, pp. 107-111.

DeMarco, T. (1993) 'Why Does Software Cost So Much?', *IEEE Software,* vol. 10, no. 2, pp. 89-90.

Dietzsch, A. (2002) 'Adapting the UML to Business Modelling's Needs – Experiences in Situational Method Engineering' in Jézéquel, J.M., Hussmann, H. and Cook, S. (eds.) *UML 2002 - The Unified Modeling Language : 5th International Conference, Dresden, Germany,* pp.73-83.

Guilford, S., Rugg, G. and Scott, N. (2002) 'Pleasure and pain: perceptual bias and its implications for software engineering', *Software, IEEE,* vol. 19, no. 3, pp. 63-69.

Jackson, M.A. (2003) 'Where, Exactly, Is Software Development?' *Formal Methods at the Crossroads: From Panacea to Foundational Support: 10th Anniversary Colloquium of UNU/IIST the International Institute for Software Technology of The United Nations University Lisbon, Portugal,* pp.115-131.

Jackson, M.A. (2001) *Problem frames: analysing and structuring software development problems,* Harlow, Addison-Wesley.

Jackson, M.A. (1995) *Software requirements & specifications: a lexicon of practice, principles and prejudices,* Harlow, Addison-Wesley.

Martin, V.A., Lycett, M. and Macredie, R. (2003) 'Exploring the Gap between Business and IT: an Information Culture Approach', *Action in Language, Organisations and Information Systems, Linköping, Sweden,* [online], http://www.vits.org/konferenser/alois2003/html/6895.pdf. [Accessed 21st April 2006].

Open University M801 (2003) *Research Project and dissertation: Study Guide,* Milton Keynes, Open University.

Open University M878/Unit 1 (2000) *Object-oriented software development: an introduction,* Milton Keynes, Open University.

Oppenheim, A.N. (2001) *Questionnaire Design, Interviewing and Attitude Measurement,* (new edn), London, Continuum.

Otero, M.C.C.M. and Dolado, J.J.C.M. (2002) 'An Initial Experimental Assessment of the Dynamic Modelling in UML', *Empirical Software Engineering,* Springer: vol. 7, no. 1, pp. 27-47.

Rapanotti, L., Hall, J.G., Jackson, M. and Nuseibeh, B. (2004) 'Architecture-driven problem decomposition' *Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04),* pp.80-89.

Rugg, G. and Petre, M. (In press) *A Gentle Guide to Research Methods,* Maidenhead, Open University Press.

Rumbaugh, J., Booch, G. and Jacobson, I. (2004) *The Unified Modeling Language reference manual, second edition,* Harlow, Addison-Wesley.

Rumbaugh, J., Booch, G. and Jacobson, I. (1999) *The Unified Modeling Language reference manual,* Harlow, Addison-Wesley.

Seamon, J.G. and Kenrick, D.T. (1994) *Psychology,* (2nd edn), Englewood Cliffs, New Jersey, Prentice Hall inc.

Sharpe, J.A., Peters, J. and Howard, K. (2002) *The management of a student research project* (3rd edn), Aldershot, Gower/Milton Keynes, The Open University.

Sikkel, K., Wieringa, R. and Engmann, R. (2000) 'A Case Base for Requirements Engineering: Problem Categories and Solution Techniques' *Proceedings of the Sixth International Workshop on Requirements Engineering: Foundation for Software Quality,* Essen, pp. 80-85.

Silverman, D. (2005) *Doing Qualitative Research,* (2nd edn), London, Sage Publications Ltd.

Thomas, D. (2005) IT budget cuts lead to rise in maintenance costs, *Computing magazine,* 20th July 2005 [online], http://www.computing.co.uk/computing/news/2140136/budget-cuts-lead-rise. [Accessed 18th March 2006].

Turk, C. and Kirkman, J. (1989) *Effective writing: Improving Scientific, Technical, and Business Communication* (2nd edn), London, E & FN Spon.

Turski, W.M. (1986) quoted in Jackson, M.A. (2003) 'Why software writing is difficult and will remain so', *Information Processing Letters,* vol. 88, pp. 13-25.

Tyndale-Biscoe, S., Sims, O., Wood, B. and Sluman, C. (2002) 'Business modelling for component systems with UML' *Proceedings of the Sixth International ENTERPRISE DISTRIBUTED OBJECT COMPUTING Conference (EDOC'02),* pp.120-131.

Wilson, M. and Sapsford, R. (2006) *Data Collection and Analysis,* in Sapsford, R. and Jupp, V. (eds) (2nd edn), London, Sage Publications Ltd.

Wolff, J.S. and Wogalter, M.S. (1998) 'Comprehension of pictorial symbols: effects of context and test method', *Human Factors,* vol. 40, no. 2, pp. 173-186.

Index

Appendix A – Business Problem Descriptions

Overview

The following pages define the business problems selected as the basis for the models used in the comprehension testing. For each problem there is a short description followed by the list of requirements modelled in each case using the UML and Problem Frames notations; it is this list of user requirements in each case that the users were required to identify from the models. For each requirement there exists a mapping to the specific UML diagram elements and to the selected Problem Frames together with the reasoning behind the decision.

Problem 1 - Description

This problem is based upon the case study in Cox et al. (2004), it relates to a financial services company, whose services include online share dealing, allowing customers to buy and sell shares. Each customer has an account, allowing them secure access to the trading area of the site where they can make purchase and sale transactions. Commission is charged to the customer based upon either a flat fee or, for higher value transactions, a percentage of the value of the transaction. A separate cash account is maintained, into which any proceeds from sale transactions are automatically credited. Purchase transactions are also funded from the same cash account which has an authorised credit limit set for each customer. Any overdraft amounts on the cash account must be settled within 10 days of the related transaction and the total overdraft cannot exceed the agreed credit limit.

A new customer can register with the site if they wish and receive e-mail news and information relating to equities trading. In order to become a trading customer they must set up a new account which will include access to the secure part of the website.

Given the nature of the business, it is important to ensure the credit worthiness of new customers before they open an account. The part of the problem being described relates to the initial registration of a new customer and, where applicable, the subsequent application for a new account. All new customers must answer a standard set of questions about their personal finances during the account application process. They are then credit scored using an external credit checking agency. It is required that this should happen in real time as part of the account setup process so that the customer can set up his account and start trading as soon as possible[1].

The credit score will be used by the system to establish the level of risk represented by each prospective customer. The risk level will be used to determine the allowable initial credit limit on the trading account. Any credit scores below an agreed threshold will require a consultation with a member of the sales team, so the account setup will be aborted and the customer notified of this immediately. All customers are automatically assigned a member of the sales team, who will be notified in this case and will take responsibility for following up the application.

Notes:

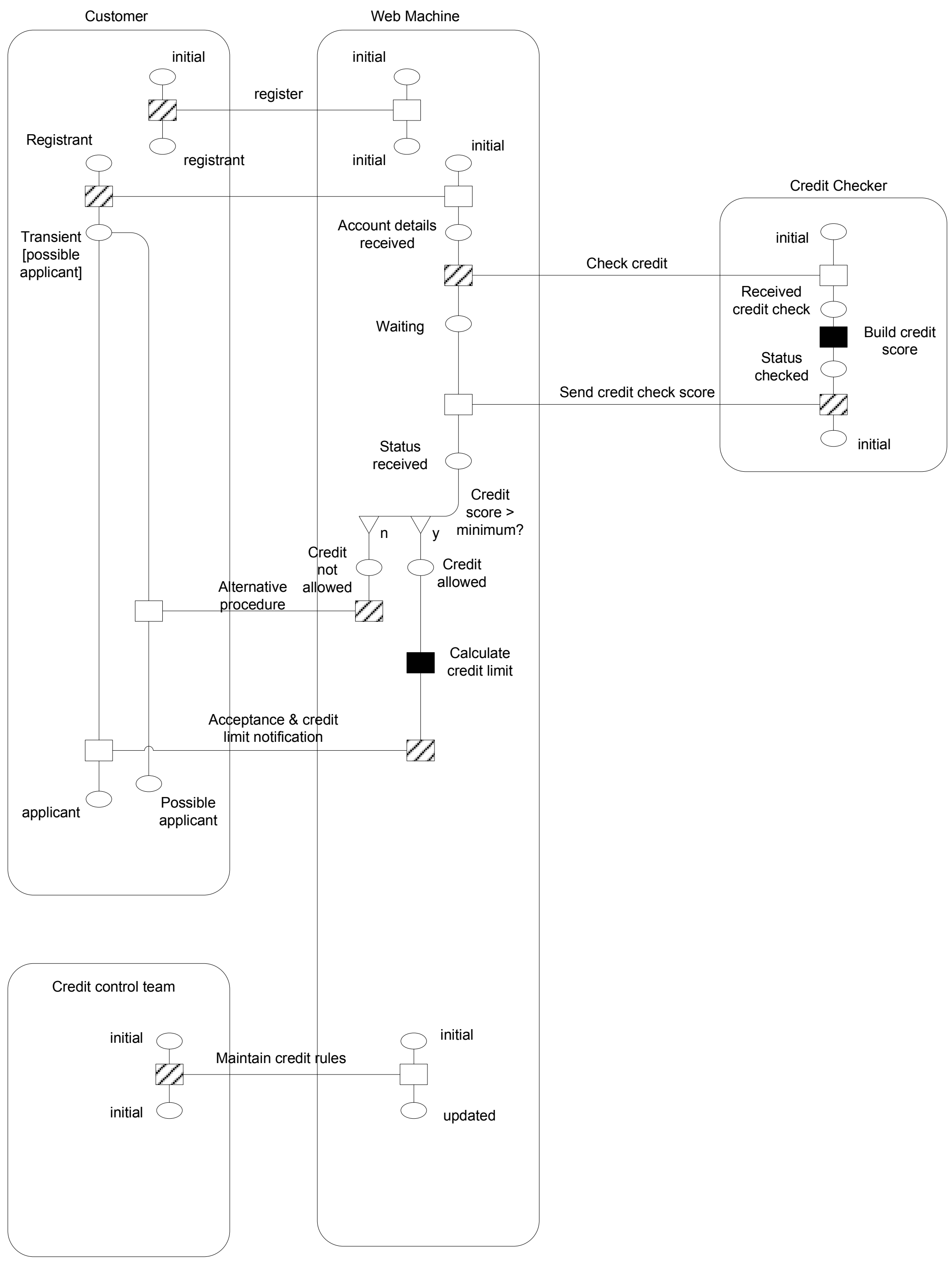
[1]This problem existed prior to recent legislation designed to prevent money laundering. Now services of this type would also include separate identity verification.

Problem 1 – User Requirements and Mapping

No.	Requirement Description	Problem Diagrams	UML
R1	A new customer must be able to register their details in order to receive e-mail updates of financial news.	**Problem Diagram 1 -** Given that the customer account resides within the machine and is maintained by a biddable domain, this seems to most closely fit a Workpieces Frame.	**Use Case:** U1-Register details, **Activity Diagram:** A1-Register details
R2	A registered customer must be able to apply for a new trading account online.	**Problem Diagram 2 -** Again this involves interaction between an entity inside the machine and a biddable domain so a Workpieces Frame seems most appropriate.	**Use Case:** U2-Apply for trading account, **Activity Diagram:** A2-Apply for trading account

No.	Requirement Description	Problem Diagrams	UML
R3	All new trading account applications must be credit scored online in real time by an external credit agency.	**Problem Diagram 3 -** This is more difficult and none of the Frames appear to fit very well here. The difficulty is in describing the information flow between the web machine and the credit check agency. If R3 and R4 are combined then the whole transaction could arguably fit a display Frame as the ultimate result is a response to the customer. This is the approach that has been taken.	**Use Case:** U3-Check credit score **Activity Diagram:** A3- Request Credit Check, A4-Calculate credit score
R4	Using the credit score, an allowable credit limit must be calculated and an immediate response given to the customer.		**Use Case:** U5-Calculate credit limit. **Activity Diagram:** A7-Calculate credit limit, A8-Create Account
R5	All credit scores below the credit limit must be referred to the sales team and the customer notified.	**Problem Diagram 4 -** This involves the passing of some information to a biddable domain (person) outside the machine, so the display Frame has been selected.	**Use Cases:** U4-Inform customer, U6-inform sales team. **Activity Diagram:** A5-Notify Sales, A6-Notify customer
R6	The credit rules shall be maintained by the credit control team.	**Problem Diagram 5 -** This involves interaction between some data held within the machine by a biddable domain so a Workpieces Frame has been selected as the most appropriate.	**Use Case:** U7**-**Maintain credit rules. **Activity Diagram:** Not represented

Problem 1 – Process Model (RAD)



Problem 2 - Background

The problem exists within the physical supply chain of a Media Company, whose main business is the creation and distribution of music. The business domain described here exists as part of the supply chain for the manufacture and distribution of physical media products - such as Compact Disc (CD) - to retail customers within Europe and the UK. Typically, when first releasing a new product such as an album the demand tends to be very difficult to predict, it usually takes at least 6 months before it settles into a more regular pattern. At this point it normally becomes possible to use sales history to accurately predict future demand using statistical forecasting techniques.

The business problem is to improve the sales forecasting of new products so that the stock can be more effectively planned. The aim is to avoid waste through overstocking while at the same time maximising availability at the point of customer order. Specifically, it is important at the initial launch of the product to get this balance right, ensuring the right amount of stock is available in the warehouse ready for shipment to retail just before the release date. For some releases, the sales drop very quickly after launch, leaving high levels of unsold stock which must be scrapped.

The specific business requirement described here is for the retail account managers and sales representatives to collaborate on a sales forecast for these new products, based upon early indications from the market, discussions with the retailers and their own experience. The resulting forecast will then be used to make decisions as to how much product to order from the manufacturer.

A software tool is needed to facilitate this collaboration and consolidate the information received into an overall forecast for a specific product. In doing this, it is important to consider smaller accounts which are not actively managed and which will therefore never have a forecast number. For these missing forecasts it is necessary to extrapolate the numbers that have been returned in order to fill the gaps. These extrapolations must be based upon very clearly defined rules which depend upon the type of product, for example classical music may have different rules to dance or pop.

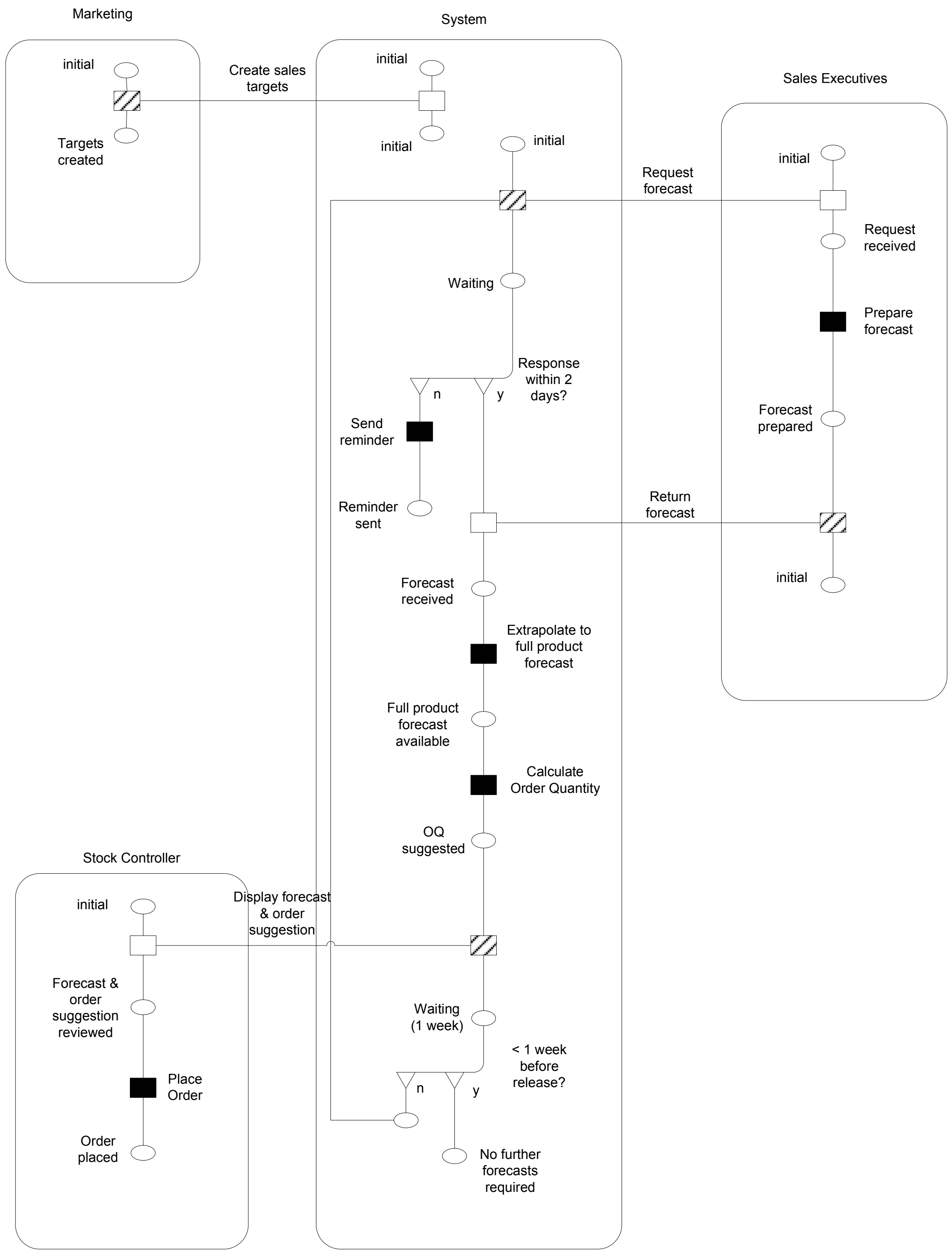
Once this extrapolation has been done, the system should consider any stock already purchased or on order and apply a standard formula to calculate the quantity to be ordered from the manufacturer.

Problem 2 – User Requirements and Mapping

No.	Requirement Description	Problem Diagrams	UML
R1	It shall be possible to maintain product sales targets for each major customer and ensure traceability to the original agreed financial budget.	**Problem Diagram 1 –** The maintenance part involves interaction between some data inside the machine and a biddable domain so a Workpieces Frame has been chosen. **Problem Diagram 2 -** The correspondence between the sales targets and the forecast is less easy to represent. It is not a required behaviour Frame because the two numbers can be different; it is just that the difference must be visible. It is therefore the display of this information which is important so a display Frame has been selected.	**Use Case:** U1-Create sales targets, U10-Review budget variance **Activity Diagram:** A1-Create sales targets
R2	A weekly e-mail shall be sent to each sales executive, requesting their input of a sales forecast for each product scheduled to be released within the next 4 weeks.	**Problem Diagram 3** - Information exchanges such as e-mail do not seem well served by the basic Frames. Cox et al, 2004 made a similar point and suggested a connection Frame for e-mail, but this focuses more on the technical aspects of e-mail systems than on the need to request information from a person. An information display seems more appropriate as the focus is on passing information (i.e. a reminder) to the sales executive.	**Use Case:** U3-Request forecast input **Activity Diagram:** A2-Request forecast
R3	If no response has been received within 2 days then a reminder shall be sent via e-mail to the sales executive.	**Problem Diagram 4** – In common with R2, the focus here is on informing a biddable domain so information display albeit via e-mail would appear to fit. This does not however capture the time element very well (i.e. within 2 days) so this	**Use Case:** U4-Send e-mail reminder **Activity Diagram:** A6 Wait 1 week, A2-Request forecast

No.	Requirement Description	Problem Diagrams	UML
		part has been included as description in the requirement oval.	
R4	The sales executive shall be able to enter a forecast of likely sales to each of their major customers.	**Problem Diagram 5 -** This involves the maintenance of a concept inside the machine by a biddable domain, so would appear to fit a Workpieces Frame.	**Use Case:** U2-Maintain forecast **Activity Diagram:** A3-Update Forecast
R5	The forecasts for each of the major customers must be extrapolated to give a full forecast based upon the relative weighting of the large customers in the market place. This weighting is maintained in a separate file.	**Problem Diagram 6 –** This is an information processing problem. Conceptually it could fit a transformation Frame - as there is an input and output - although the result will remain within the machine for use in R6 and R7. As none of the other Frames appears to fit, this is the approach adopted. The output is therefore shown as a designed domain while the input is a given domain.	**Use Case:** U6-Extrapolate to full forecast **Activity Diagram:** A4-Extrapolate to full forecast
R6	Based upon the consolidated forecast and any pre-ordered stock, the system must calculate the suggested order quantity.	**Problem Diagram 7** - Some information processing is occurring, which, although it is happening inside the machine, is a specific customer requirement so it should be modelled. As with R5, the chosen Frame is a transformation Frame but this time converting two inputs into one output.	**Use Case:** U7-Calculate order quantity **Activity Diagram:** A5-Calculate OQ
R7	The stock controller shall be able to review the number of days' stock coverage based upon the consolidated forecast and any extrapolations applied.	**Problem Diagram 8 -** This involves external representation of information to a biddable domain, so an information display Frame is the chosen approach, but with input from two other domains.	**Use Case:** U8-Review stock coverage **Activity Diagram:** Not represented

Problem 2 – Process Model (RAD)



Problem 3 - Background

This problem exists within a specialist lending library which lends a range of items such as books, CDs and DVDs to registered members. The items for loan fall into one of two categories: standard items which tend to be of lower value and easily obtained; and premium items which are much higher value and relatively rare.

Similarly there are two levels of registered membership: standard and privileged. Standard members can borrow only the standard range of items for up to 30 days, then they must return or renew in person. A privileged member can borrow the full item range and has the option to renew by telephone, or online if the item has not been reserved for another member. Privileged membership is attained through a combination of the length of membership and the number of items borrowed and returned on time. In other words, very active members who return items on time are rewarded with the additional benefits.

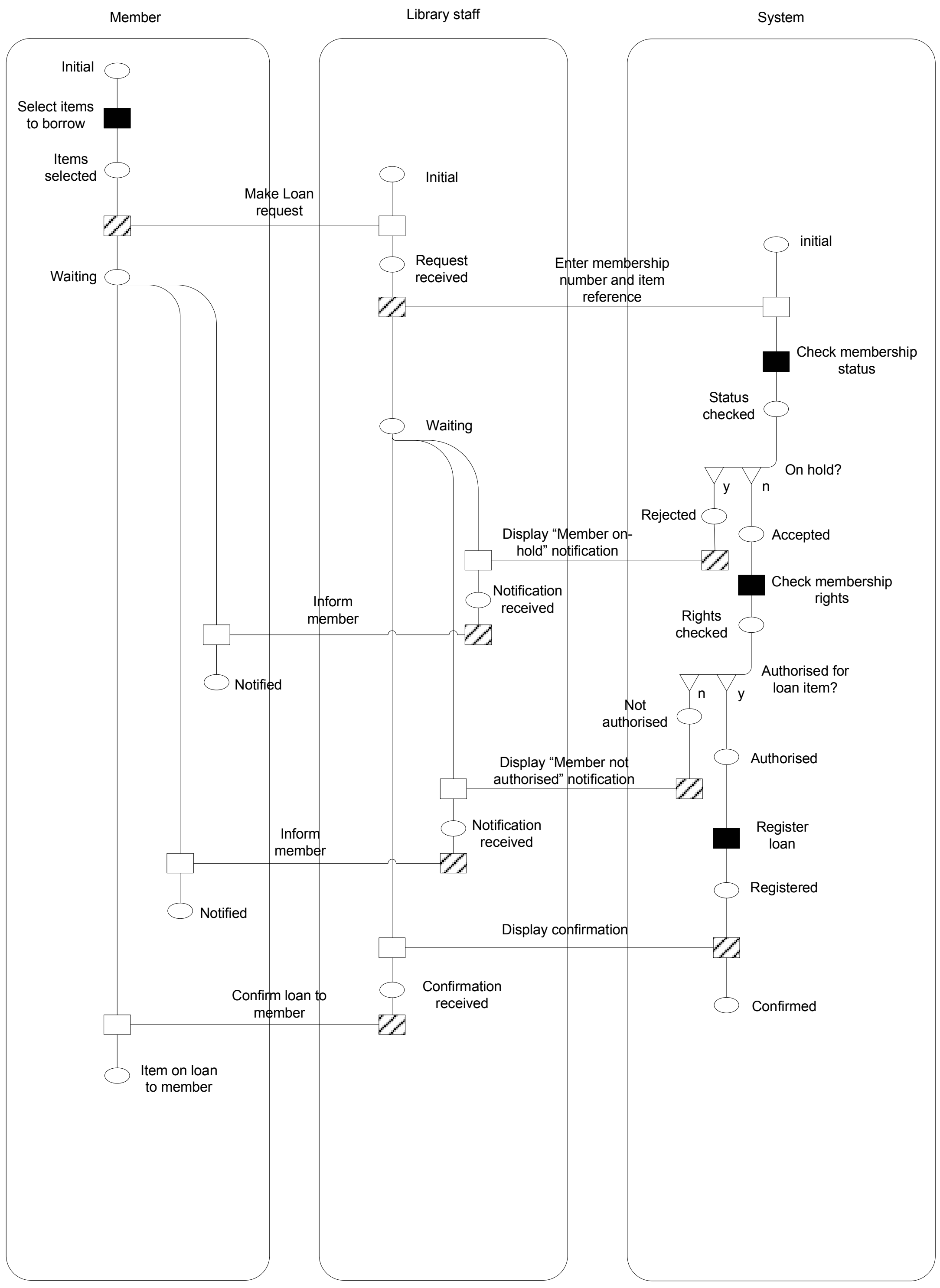
Any member can borrow up to 3 items at any time and must return them within the specified time limit. If any items are not returned or renewed within the specified period then the members account is placed on hold pending return of the items and payment of a fine. In the case of the privileged member this will also generate a black mark which can result in withdrawal of privilege status.

The business problem is to build a system to manage the lending of items to ensure that these membership rules are strictly enforced. The system will be used by cashiers within the library to administer the loan. Before the loan is registered a check must be made by the system to see if the member is on hold and has sufficient privilege rights for the item to be borrowed. If the checks fail then the loan must be refused and the cashier notified of the reasons why.

Problem 3 – User Requirements and Mapping

No.	Requirement Description	Problem Diagrams	UML
R1	A member of the library staff should be able to request a loan on behalf of a member by entering their membership number and the ID number of the item to be borrowed.	**Problem Diagram 1** – This involves the creation of a loan (if authorised) which is a persistent piece of computer data linking "item" and "member". The management of this data by an external biddable domain would indicate a Workpieces Frame.	**Use Case:** U2-Request loan **Activity Diagram:** A1-Request Loan
R2	Any members who have more than 3 items on loan should not be able to borrow more items.	**Problem Diagram 2** – This seems to be a straightforward computing rule constraining an abstraction of the items and members domains. A required behaviour Frame would therefore seem to fit this type of problem.	**Use Case:** U3-Check membership status, U6-Refuse Loan **Activity Diagram:** A3-Check membership status **Class Diagram:** Relationship constraint between Member and Loan Item.
R3	If any items are unreturned by the loan expiry date then the member will be placed on hold till the items are returned and any penalties paid.	**Problem Diagram 3** – As above, this seems to be a required behaviour problem as it is a simple rule which must be enforced on the abstracted items and members domain.	**Use Case:** U7-Monitor Loans, U8-Place member on hold.
R4	Privileged members are able to borrow any item in the library while standard members can only borrow the standard range.	**Problem Diagram 4** – Once again this seems to be another required behaviour Frame to enforce the specified lending rule. R2 to R4 could possibly have been taken together as one Frame but the chosen approach more clearly maps the individual customer requirements.	**Use Case:** U4-Check member rights **Activity Diagram:** A5-Check membership rights **Class Diagram:** Relationship between Privilege Member and Premium Loan Item.
R5	If a loan cannot be authorised, the library staff member should be notified in real time with the reason for the rejection, otherwise the transaction should be confirmed.	**Problem Diagram 5** – An information display Frame has been selected to represent this as this is about passing information to a biddable domain.	**Use Case:** U5-Confirm loan, U6-Refuse Loan **Activity Diagram:** A6-Register loan, A8-Refusal Message

Problem 3 – Process Model (RAD)



Problem 4 - Background

The problem described relates to an archive storage company which operates an archive facility for registered customers. This facility is geared mainly towards companies who have limited space but are required to store documents or other media such as backup tapes for legal or tax purposes. The company uses a range of standard sized boxes which are shipped in flat pack form to the customer. A set of labels with unique identifiers are printed on the customer's printer together with the shipment documentation. The customer then adds the items to be archived into the cartons, adds the labels to the outside then arranges with the company to have the boxes collected.

A registered customer must be able to specify a contents list together with the size and weight of the contents using an online facility. The system will use the size and weight details to assign one or more standard boxes of different sizes and generate the unique storage identifiers. The system will use aim to ensure that maximum use is made of valuable storage space.

Once the details have been entered by the customer and confirmed, the system will print a set of labels and shipping paperwork for the customer to use when he receives the boxes. It will also generate a set of paperwork for use by the internal staff which will include: a packing list showing the number and sizes of boxes to be shipped; and a despatch note with customer address details. The boxes are then packed and shipped to the customer for them to fill and return.

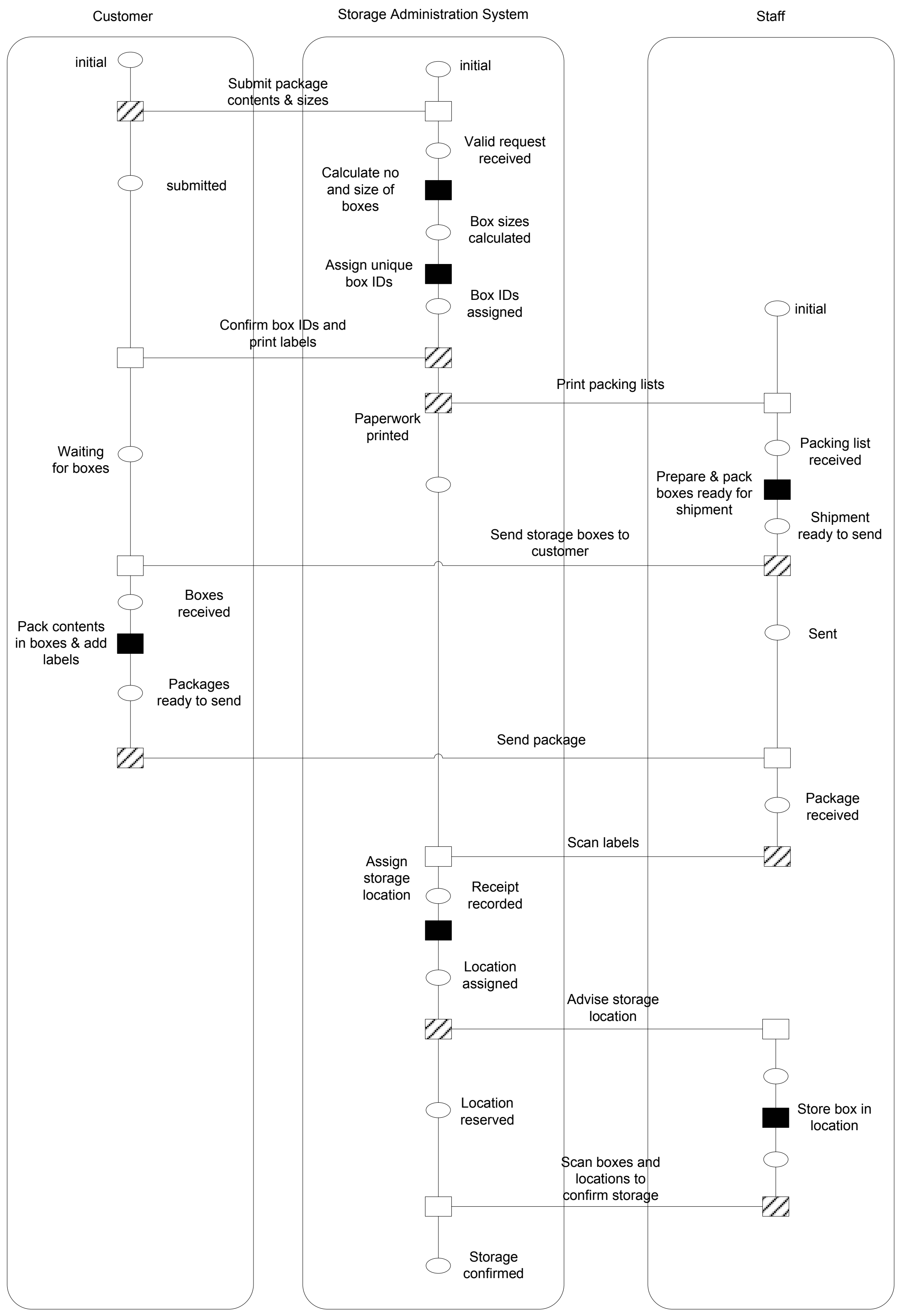
Once the filled boxes are returned, the staff will scan the bar-coded unique identifier on the carton and the system will then advise a free location where the item can be stored. As the item is placed on the shelf, the warehouse staff will scan the location and the box unique identifier to confirm where the box has been stored. This will confirm the storage transaction and update the warehouse storage location records.

Problem 4 – User Requirements and Mapping

No.	Requirement Description	Problem Diagrams	UML
R1	A registered customer should be able to create a new storage record online.	**Problem Diagram 1 –** This involves interaction between a biddable domain and a concept inside the machine so a Workpieces Frame has been selected.	**Use Case:** U1-Create storage record **Activity Diagram:**
R2	When entering contents and overall package sizes the system should calculate the most appropriate box sizes and number of boxes to minimise storage space in the warehouse. A unique box ID will be assigned for each box.	**Problem Diagram 2 –** This is a data processing activity receiving inputs and giving outputs so a Transformation Frame seems most appropriate here. It could be argued that this is implementation detail but for completeness in describing the requirement it has been included.	**Use Case:** U2-Submit package contents & sizes, U3-Generate unique box IDs **Activity Diagram:** A1-Submit package contents & sizes, A2-Calculate qty & sizes of boxes
R3	The labels and shipping documentation will be printed on the customer’s printer once the transaction is confirmed.	**Problem Diagram 3 –** Printing does not seem well covered in the PF approach. It could theoretically be a Display Frame if one considers printed output to be a display. Alternatively it could be a transformation Frame converting computer file data into printed output. Neither fits the requirement that well but the information display seems intuitively closer because it involves passing information to a biddable domain.	**Use Case:** U4-Print paperwork **Activity Diagram:** A3-Print Paperwork
R4	Meanwhile a packing list and despatch note must also be printed on the warehouse printer.	**Problem Diagram 4 –** For the same reasons listed in R3 an information display Frame has also been selected for this requirement.	**Use Case:** As R3 **Activity Diagram:** As R3
R5	On receipt of the filled boxes the warehouse staff will scan the barcode labels using a remote data terminal and receive a storage location advice on the screen.	**Problem Diagram 5 –** Given that this problem involves interaction between a biddable domain and an internal file, a Workpieces Frame could be selected. However the important thing is the advice of the put-away location so an Information Display Frame seems more appropriate. In this case it is important to capture the domains which contribute to the result which means that a composite Frame seems more appropriate.	**Use Case:** U5-Scan box labels, U6-Assign storage location **Activity Diagram:** A7-Scan labels, A8-Assign storage location
R6	Once the items have been stored, the warehouse staff will scan both the location barcode and the box ID to confirm the	**Problem Diagram 6 –**This ultimately involves interaction between a biddable domain and an internal file (to update the storage	**Use Case:** U7-Confirm storage location **Activity Diagram:** A11

No.	Requirement Description	Problem Diagrams	UML
	storage.	record) so that would suggest a Workpieces Frame. As with R5, the interaction involves a number of domains so it has been modelled as a composite problem.	Confirm location, A12-Update storage records

Problem 4 – Process Model (RAD)



Appendix B – Introductory Note Sent to Participants

Preamble & Research Background

Thank you for agreeing to be part of this process, I will try not to take too much of your time. I am carrying out some research for a dissertation as part of a Master's degree in Computing for Commerce and Industry with the Open University.

The dissertation will discuss the communication that takes place within typical software projects between software specialists and business stakeholders. The research will attempt to compare two methods proposed within the IT literature for domain modelling and requirements analysis. The term "domain modelling and requirements analysis" refers to the detailed description of what a system must do and the environment (domain) in which it will be expected to operate.

Some research I have found suggests that, on average, IT projects have a 29 per cent success rate, with an average cost overrun of 59 per cent and schedule delay of 84 per cent[1]. In terms of achieving the required benefits, 31% of systems built are never delivered and another 15% have satisfied fewer than half the intended customers' needs[2]. A failure in the requirements analysis activity is regularly cited as one of the major contributors to these statistics.

The main objective of requirements analysis and domain modelling is to achieve consensus among all stakeholders on the details of what needs to be built. It requires communication between those who will build the software and those who will use or be affected by it. As the statistics show, this is easier in theory than in practice and many methods have been developed over the years to try to increase the effectiveness of this activity.

I have chosen two such methods as the basis for this research, being among the more prominent within current software literature: the Unified Modelling language (UML) and Problem Frames. An overview of each has been included below, together with a brief introduction to the notations.

Your Contribution to the Research Process

Four simple software problems have been selected and described separately using each of the methods. During the interview you will be asked to review two problems using each method and describe to me what you understand about the problem as presented.

I must stress at this point that this is not in any way a test of your understanding of IT, general intellect, experience or ability. I have specifically chosen only participants with little or no previous experience of either method. This is because I am interested in the extent to which the models easily convey important information to readers with little or no prior knowledge of the notation.

An Overview of the Notations

1 – The Unified Modelling Language (UML)

The UML was created from a number of methods, most of which evolved to support the Objected Oriented (OO) programming paradigm. It was eventually adopted as a formal standard in 1997, managed by the Object Management Group (OMG) and has a good deal of supporting literature. However its use within commercial organisations remains patchy, with many organisations adopting only certain aspects or using it only within specific projects.

The UML standard[3] includes support for domain modelling by capturing aspects of the business environment in terms normally recognised by the business users. This is achieved by focusing upon the key concepts (objects) existing in the real world and using those as the basis for discussion. Examples of these concepts would be orders, products and invoices and customers.

There are three main diagrams proposed within the UML for business domain modelling:

Use case models – These define the interactions with the system or business domain by external entities or "actors". The actors would typically be people or other systems and are represented by stick figures; use cases are interactions with the system and are represented by an oval containing a short description of the use case. A line connecting the ovals to the actors indicates the relationship between an actor and one or more use cases.

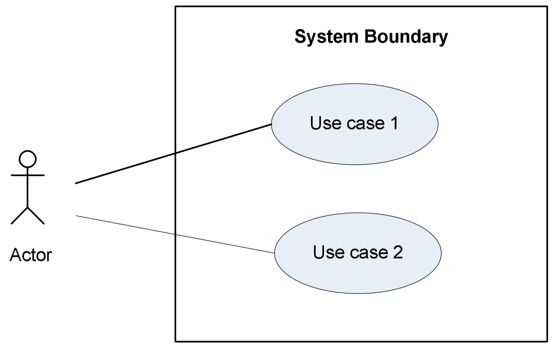


Figure 1 - Use Case Diagram

Activity diagrams – These model the process flows existing within the business domain or system. They follow a similar notation to more traditional swim lane models, with processes being represented by rounded boxes and outputs as square boxes. A diamond is used to indicate a branch or decision.

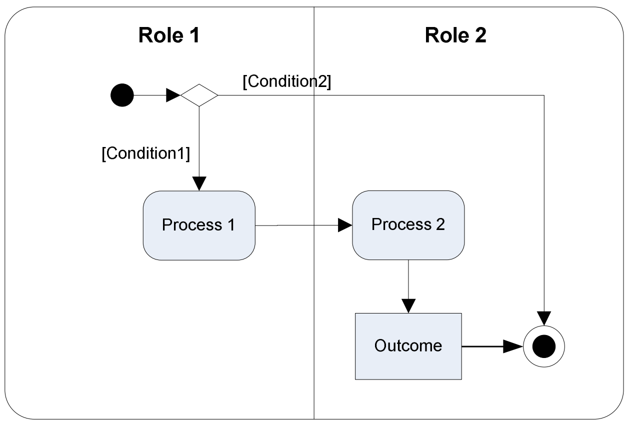


Figure 2 - Activity Diagram

Class / concept diagrams – These model the main concepts existing in the business domain and the relationships between them. They are represented by square boxes connected by lines indicating a relationship between the concepts, with the direction indicated by a small arrow. An important part of the relationship is to understand the relative number of instances of one concept which would relate to the other in the relationship. As an example, consider an order and order line relationship; each order would normally have at least one or possibly more order lines. This idea is known as multiplicity and it is shown at each end of the relationship as a number range (such as one to three) or an asterisk, which means any number including zero.

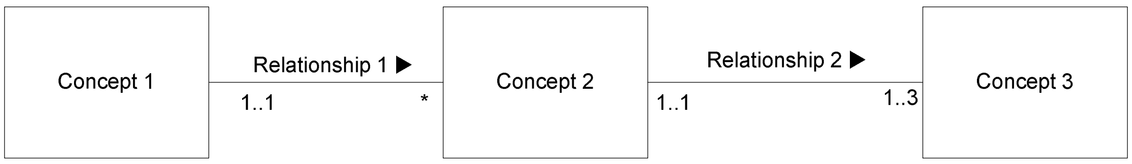


Figure 3 - Class Diagram

2 – Problem Frames

Problem Frames are a more recent development and have been gaining increasing attention within academic literature. They were developed by Michael Jackson, whose Jackson Structured Design (JSD) and Jackson Structured Programming (JSP) methods were developed in the early 1980s and eventually formed part of the Systems Analysis and Design Method (SSADM). SSADM was adopted as a government standard for software development projects and is still widely used today. Given that Problem Frames are a more recent development, only a few examples of its use exist currently in the commercial world.

Problem Frames and the Problem Diagram notation focus upon the problem environment and the observable changes which should be evident in the real world once the software is implemented. Jackson[4] suggests that existing methodologies tend to focus primarily on the system boundary, i.e.

inputs and outputs, while some important details of the problem are deeper into the real world. Ultimately it is the desired observable change in the world that is important to the customer, so software design should take into account all needs to happen to ensure the change occurs correctly.

The graphic notation used in problem diagrams is relatively simple. Each diagram includes the following components:

- **Domains** – These are things which exist in the real world and are observable; some examples are: people, other systems, mechanical devices or software files. Domains are normally represented by simple rectangular boxes.
- **Designed Domains** – A Designed domain is a computer file which is important to the problem and evident in the real world, but the software designer is free to decide on its precise format. It is indicated by a rectangle with a single stripe down the left hand side.
- **Machine** – This is a general purpose computer, whose software will cause the observable effects to occur in the real world. It is represented by a rectangle with a double stripe down the left hand side.
- **Requirement** – This is essentially a constraint on one of the domains (indicated by the dotted arrow). It is a statement indicating what must be true if the software is to be considered a success. The graphic symbol used is an oval with a dotted line.
- **Interfaces** – These are shared phenomena, which exist between the other elements of the problem. In Figure 5 below, interfaces E and F indicate which phenomena the machine will use to interact with the domain; while the dotted lines G and H indicate the phenomena of the domain that can be observed in the real world to confirm the requirement has been met. The arrow head indicates which domain is to be controlled by the machine in some way. The other domains are therefore given and therefore cannot be changed or affected by the machine.

There are two principle sets of diagrams used within the Problem Frames approach:

Context diagrams – These define where the new software will operate and which domains are affected. They show the domains and the interconnections between them and the software.

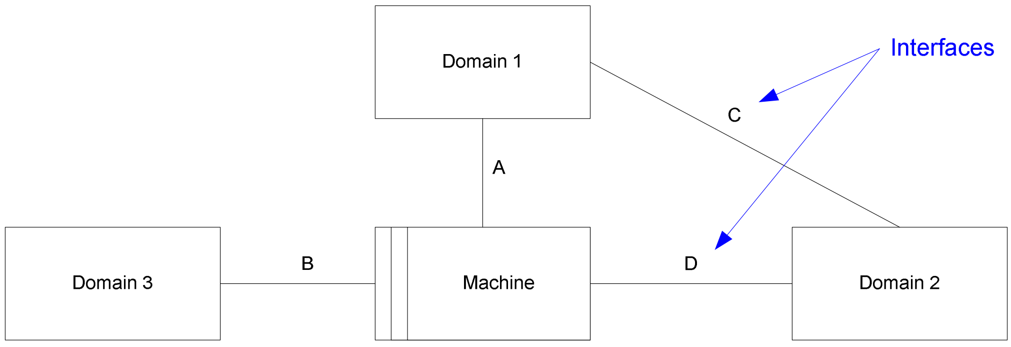
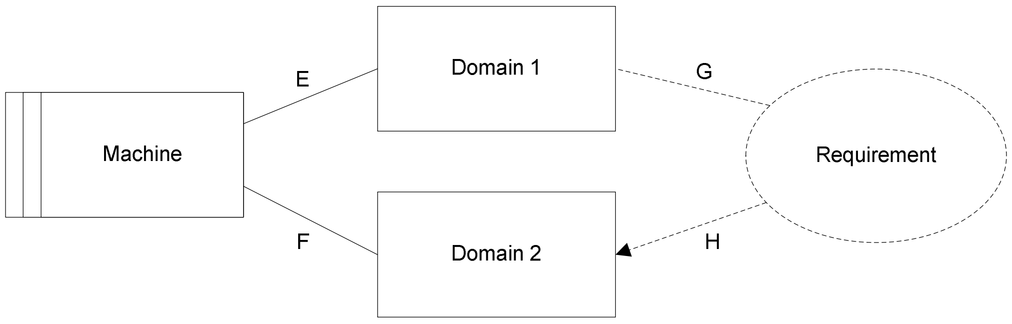


Figure 4 - Problem Context Diagram

Problem diagrams – These reflect the desired situation in terms of the observable effect on the domains in the problem.



Interfaces:

Interface	Description
E	D1! {Domain 1 phenomena captured by the machine},
F	MA! {Domain 2 phenomena controlled by the machine}
G	D1! {Observable given phenomena about Domain 1}
H	D2! {Observable desired phenomena about Domain 2}

Indicates which domain is leading in the phenomena exchange (interface F is the machine (MA) and H is Domain 2 (D2))

Figure 5 - Problem Diagram

References

1. Thomas, D. (2005) IT budget cuts lead to rise in maintenance costs, *Computing magazine,* 20th July 2005 Found at: *http://www.computing.co.uk/computing/news/2140136/budget-cuts-lead-rise.*

2. Davis, A.M. and Hickey, A.M. (2002) 'Requirements Researchers: Do We Practice What We Preach?', *Requirements Engineering,* vol. 7, no. 2, pp. 107-111.

3. Rumbaugh, J., Booch, G. and Jacobson, I. (2004) *The Unified Modeling Language reference manual, second edition,* Second edn. Reading, MA; Harlow, Addison-Wesley.

4. Jackson, M.A. (2001) *Problem frames: analysing and structuring software development problems,* Harlow, Addison-Wesley.

Appendix C – Interview Script (Questionnaire)

Section 1 – Your Background

For each question below, please select your chosen answer by circling the number to the left of it. In some cases you should choose only one and in others you may be asked to select all that apply.

This section aims to understand a little about your previous exposure to software related projects and the typical role you would have played in them.

Q 1. Please indicate the number of software projects to which you have had some exposure in the past. (Circle one only):

1. 0.
2. 1 – 2.
3. 2 – 5.
4. 5 – 10.
5. More than 10.

☞ *Note: Exposure as defined here means you would either have been directly involved (for example managing the project or agreeing the requirements) or directly affected by the outcome, maybe as a manager of a business unit or department.*

Q 2. Please select which of the following best describes your most common role in software related projects. (Circle one only):

1. Business / Systems Analyst.
2. Business manager with a stake in the outcome.
3. Project owner.
4. Business area (domain) expert.
5. User.
6. Other – Please state below:

____

Q 3. Have you ever been asked to approve software requirements specifications and if so in what form were they? (Please circle the one that applies most):

1. Never been asked to sign off a requirements document.
2. They were mostly text descriptions, with a few supporting diagrams.
3. They were mostly diagrams but with some supporting text.
4. I have approved both text based and diagram based documents in the past.

Feel free to comment further here if required:

Q 4. In general and based upon your previous experience, which statement best reflects how you prefer to review complex descriptions (whether software or any other business requirement):

1. I generally prefer text descriptions.
2. I generally prefer diagrams.
3. I have no preference.

Feel free to comment further here if required:

Q 5. Have you ever been involved in the development of models using any of the notations listed below? (Please circle all that apply or choose option 6):

1. Data Models / Entity Relationship Diagrams (ERD).
2. Data Flow Diagrams (DFD).
3. State Machines / Charts.
4. Role Activity Diagrams (RAD).
5. “Swim-lane” style process models.
6. No development experience of any of the above.

Feel free to comment further here if required:

Q 6. Have you ever knowingly been asked to review models using any of the notations listed below? (Please circle all that apply or choose option 6):

☞ *Note: Only select those you were consciously aware of at the time, i.e. you know what they were.*

1. Entity Relationship Diagrams (ERD).
2. Data Flow Diagrams (DFD).
3. State Machines / Charts.
4. Role Activity Diagrams (RAD).
5. "Swim-lane" style process models.
6. No experience of any of the above that I am aware of.

Feel free to comment further here if required:

Q 7. Have you ever been involved in the development of models using the notations we will be using today? (Please circle all that apply or choose option 6):

1. UML Use Case Diagrams.
2. UML Activity Diagrams.
3. UML Class Diagrams.
4. Problem Frame Context Diagrams.
5. Problem Frames / Problem Diagrams.
6. None that I am aware of.

Feel free to comment further here if required:

Q 8. Have you ever been asked to review models using any of the notations we will be using today? (Please circle all that apply or choose option 6):

1. UML Use Case Diagrams.
2. UML Activity Diagrams.
3. UML Class Diagrams.
4. Problem Frames Context diagrams.
5. Problem Frames / Problem Diagrams.
6. None that I am aware of.

Feel free to comment further here if required:

Review of example software models

The following section is the core part of the research and will require you to review four sets of models describing four different software problems. For each model you are asked to list the software requirements as they are conveyed to you by the models.

Once again I must stress that this is not in any way a test of your abilities or experience. The aim is to build a picture over a sample group of the relative strengths of each model in conveying the important aspects of typical commercial software problems.

Section 2 – Software Model Review

Problem 1

Q 9. Please review the models given to you and list, as fully as possible, the requirements as conveyed in the diagrams.

No.	Description
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Q 10. Please indicate the level of difficulty experienced in understanding these models (circle the number which best indicates the level of difficulty):

Very easy				Very confusing
5	4	3	2	1

Problem 2

Q 11. Please review the models given to you for problem 2 and list, as fully as possible, the requirements as conveyed in the diagrams.

No.	Description
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Q 12. Please indicate the level of difficulty experienced in understanding these models (circle the number which best indicates the level of difficulty):

Very easy				Very confusing
5	4	3	2	1

Problem 3

Q 13. Please review the models given to you for problem 3 and list, as fully as possible, the requirements as conveyed in the diagrams.

No.	Description
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Q 14. Please indicate the level of difficulty experienced in understanding these models (circle the number which best indicates the level of difficulty):

Very easy				Very confusing
5	4	3	2	1

Problem 4

Q 15. Please review the models given to you for problem 4 and list, as fully as possible, the requirements as conveyed in the diagrams.

No.	Description
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Q 16. Please indicate the level of difficulty experienced in understanding these models (circle the number which best indicates the level of difficulty):

Very easy				Very confusing
5	4	3	2	1

Section 3 – Your feedback

Q 17. On reflection, did you recognise any of the models from previous projects or any literature you have read? (Please circle all that seemed familiar or choose 6 if none):

1. UML Use Case Diagrams.
2. UML Activity Diagrams.
3. UML Class Diagrams.
4. Problem Frame Context Diagrams.
5. Problem Frames / Problem Diagrams.
6. None recognised.

Feel free to comment further here if required:

Q 18. Overall, please indicate which set of models you felt was the easiest to understand (please circle one only):

1. UML.
2. Problem Frames.
3. Both were equally difficult.
4. Both were equally easy to understand.

Feel free to comment further here if required:

Q 19. Please indicate which, if any of the specific diagrams you preferred (please circle one only), then indicate below your reasons why:

1. UML Use Case Diagrams.
2. UML Activity Diagrams.
3. UML Class Diagrams.
4. Problem Frames Context Diagrams.
5. Problem Frames / Problem Diagrams.
6. No particular preference.

Reasons:

Q 20. Please reflect on the last requirements document you approved (which I accept may be some time ago), then try to remember honestly how you felt when reviewing the document (circle the statement that most closely describes your reaction):

1. Not applicable – I have never approved a software requirements document.
2. I felt very comfortable that the requirements were clearly and completely documented, both in summary and in the detail.
3. Felt ok in general but didn't fully understand all the finer detail, that's up to the developers.
4. I actually understood very little but I'll just hold the project team accountable if it is wrong!
5. Something else – Please comment below:

Comment:

Q 21. Comparing the models used in this research with other requirements documents you have reviewed in the past, please indicate how they rate in your opinion (please circle one only):

1. The UML notation was an improvement but the Problem Frames was not.
2. The Problem Frames notation was an improvement but the UML was not.
3. Both were an improvement.
4. Neither was an improvement.
5. Unable to answer as I have nothing to compare against.

Feel free to comment further here if required:

If you have any further comments you would like to add regarding the notations used or the research in general, please feel free to enter them in the box provided below:

Many thanks!

I would like to thank you very much for your time completing this exercise and helping me with my research.

Confidentiality

All responses given will remain anonymous as far as individuals are concerned and no names will be mentioned within the finished dissertation; the information collected will be used to build an overall picture. However, I will be happy to acknowledge your input within the final paper if you so wish and you are also welcome to receive a copy of the finished dissertation, probably available mid 2007.

Please indicate below your preferences by ticking the boxes provided ☑:

☐ **I would like to be acknowledged by name in the final dissertation.**

☐ **I would like to receive a copy of the final dissertation.**

Your Details (optional):

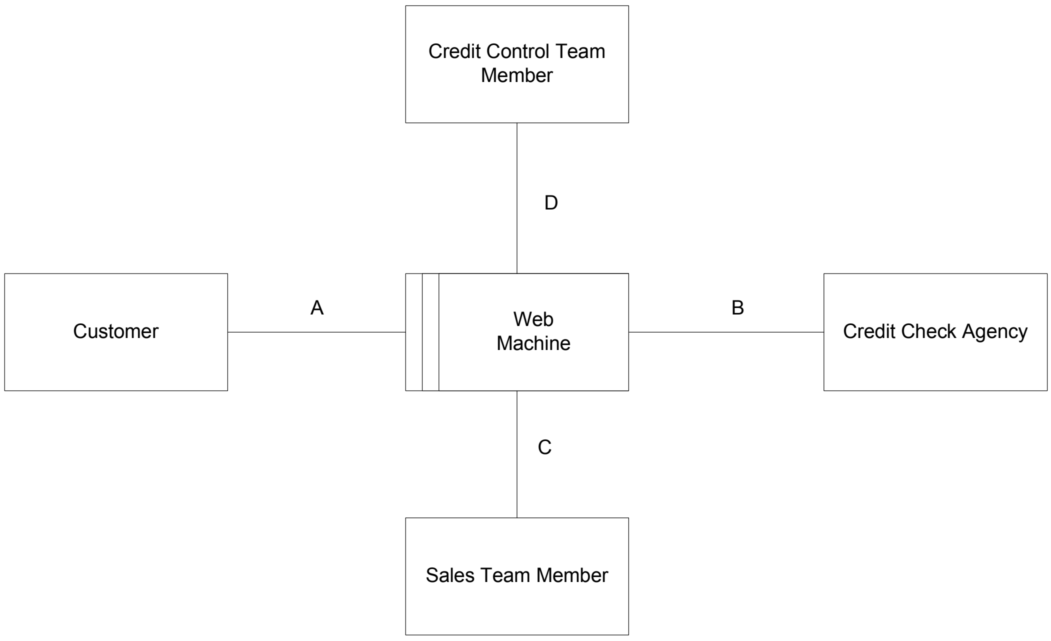
Name:

Company:

Job Title:

Appendix D – The UML and Problem Frames Models

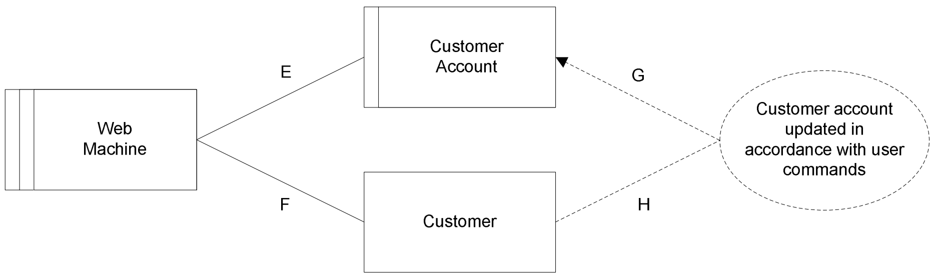
Problem 1 - Context



Interfaces:

Interface	Description
A	Account management
B	Customer details, credit answer
C	Failed credit checks
D	Credit rules

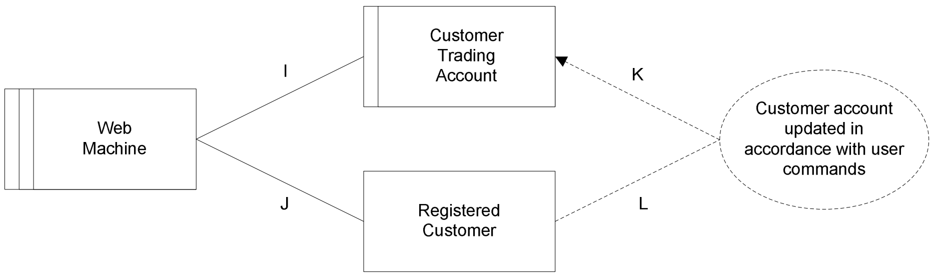
Problem 1 – Diagram 1



Interfaces:

Interface	Description
E	WM! {Account edit operations}, CA! {Customer account states}
F	CU! {Editing commands}
G	CA! {Customer account effects}
H	CU! {User commands}

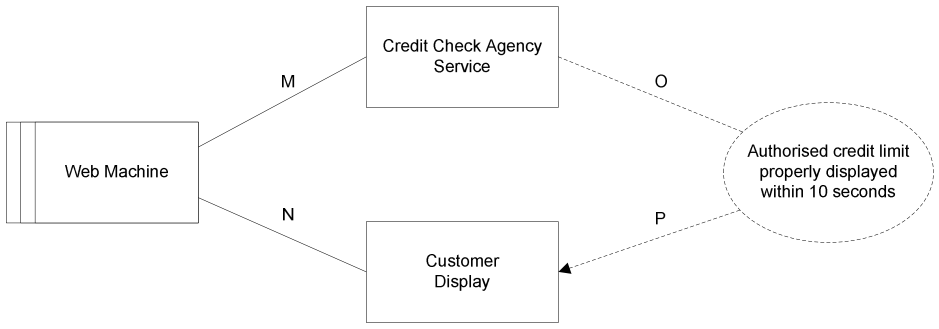
Problem 1 – Diagram 2



Interfaces:

Interface	Description
I	WM! {Account edit operations}, CA! {Account states}
J	RC! {Editing commands}
K	CA! {Customer trading account effects}
L	RC! {User commands}

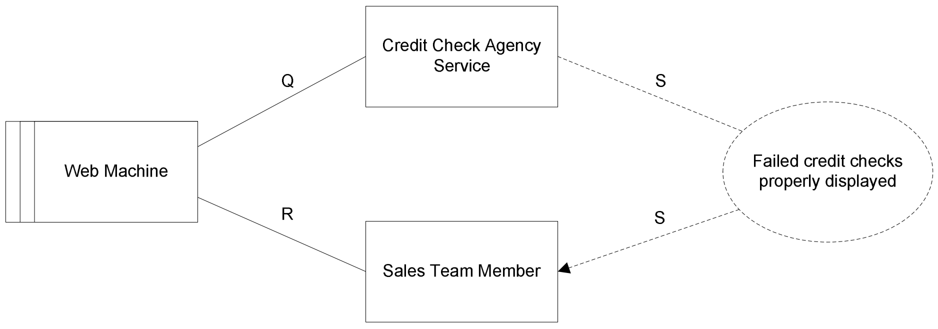
Problem 1 – Diagram 3


Credit Check Agency Service
Web Machine
Customer Display
Authorised credit limit properly displayed within 10 seconds
M
N
O
P

Interfaces:

Interface	Description
M	WM! {Formatted request message}, CC! {Formatted response}
N	WM! {Credit check result}
O	CC! {Credit score}
P	WM! {Allowable credit}

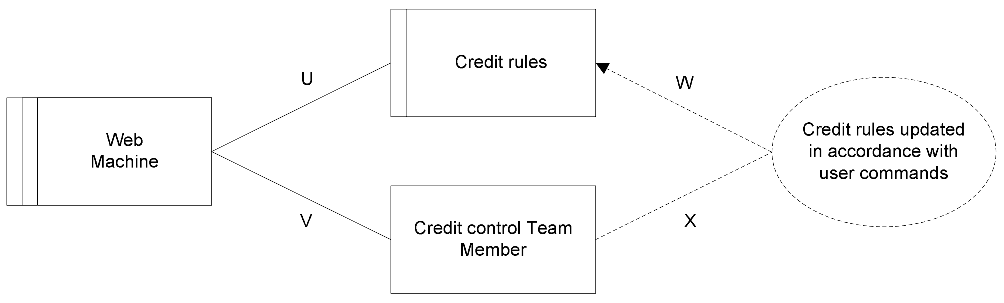
Problem 1 - Diagram 4


Credit Check Agency Service
Web Machine
Sales Team Member
Failed credit checks properly displayed
Q
R
S
S

Interfaces:

Interface	Description
Q	WM! {Formatted request message}, CC! {Formatted response}
R	WM! {Low scoring applications}
S	WM! {Customer details}, CC! {Credit score}
T	WM! {Low scoring applications}

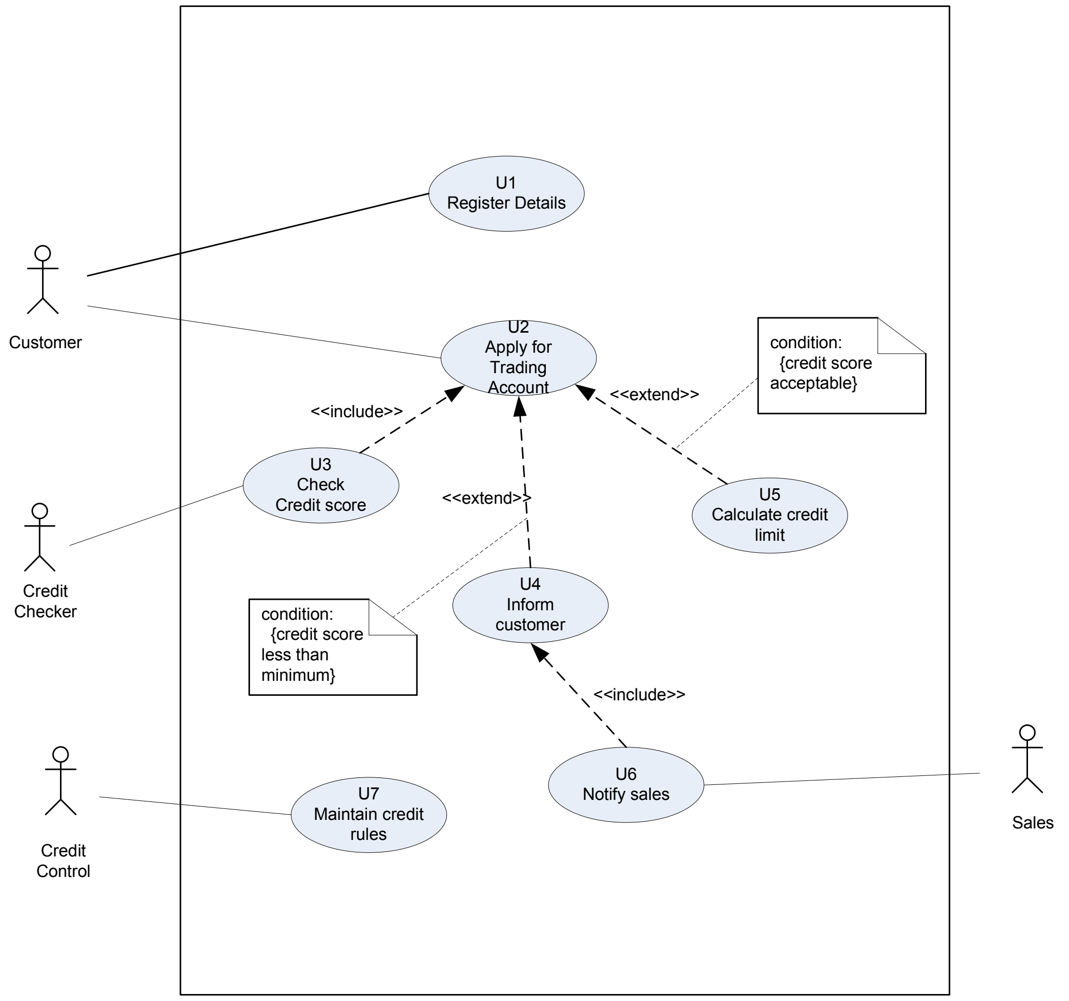
Problem 1 – Diagram 5


Web Machine
Credit rules
Credit control Team Member
Credit rules updated in accordance with user commands
U
V
W
X

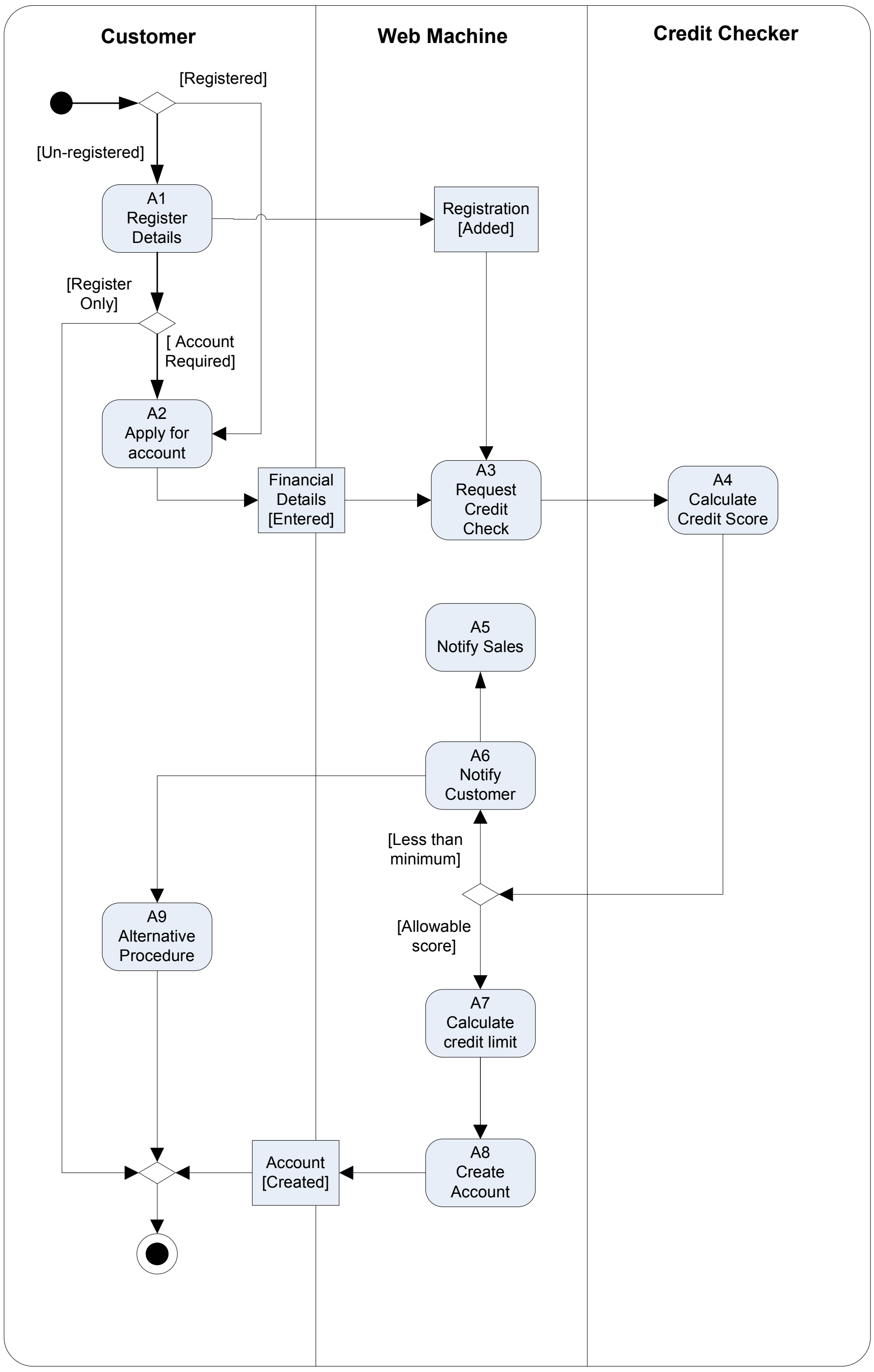
Interfaces:

Interface	Description
U	WM! {Rules editing operations}, CR! {Rule states}
V	CC! {Editing commands}
W	CR! {Credit rules effects}
X	CC! {User commands}

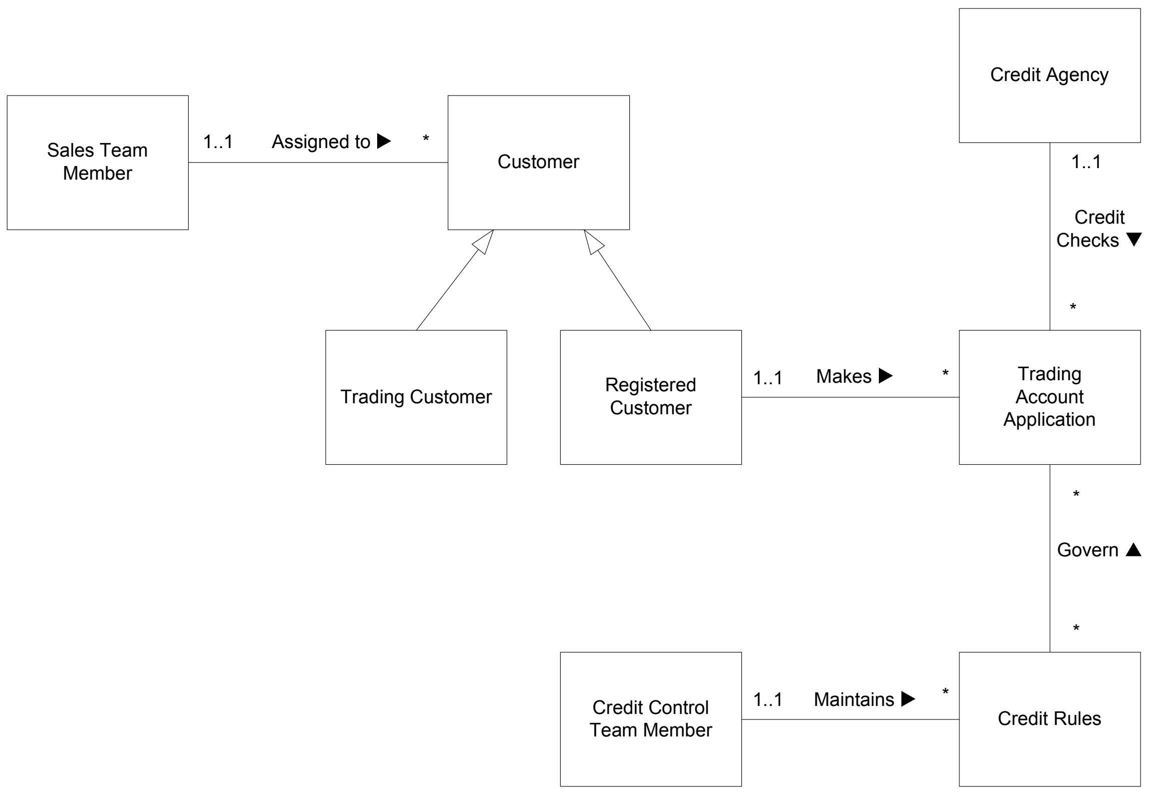
Problem 1 - Use Case View



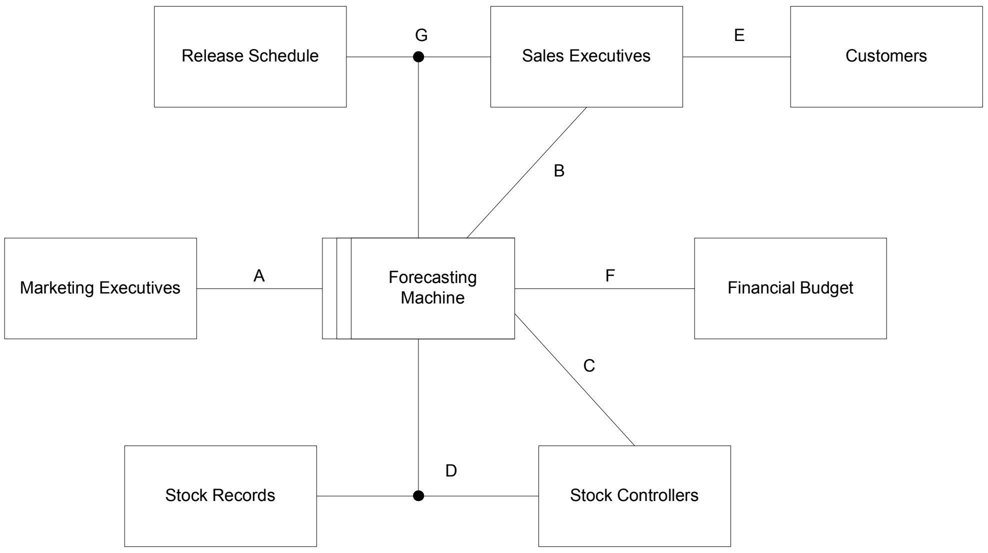
Problem 1- Activity Diagram



Problem 1- Class Diagram – Domain concepts



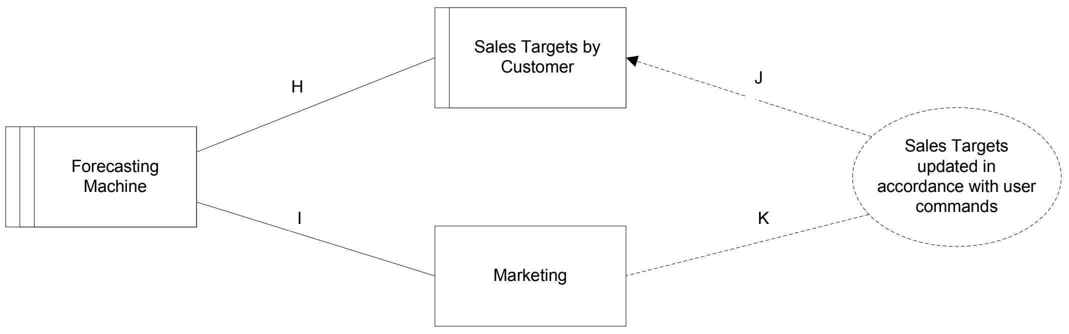
Problem 2 - Context



Interfaces:

Interface	Description
A	Sales targets
B	Sales forecast request, forecast response
C	Consolidated forecast, Suggested Order Qty
D	Current inventory levels
E	Estimated consumer demand
F	Budget quantity
G	Scheduled release date

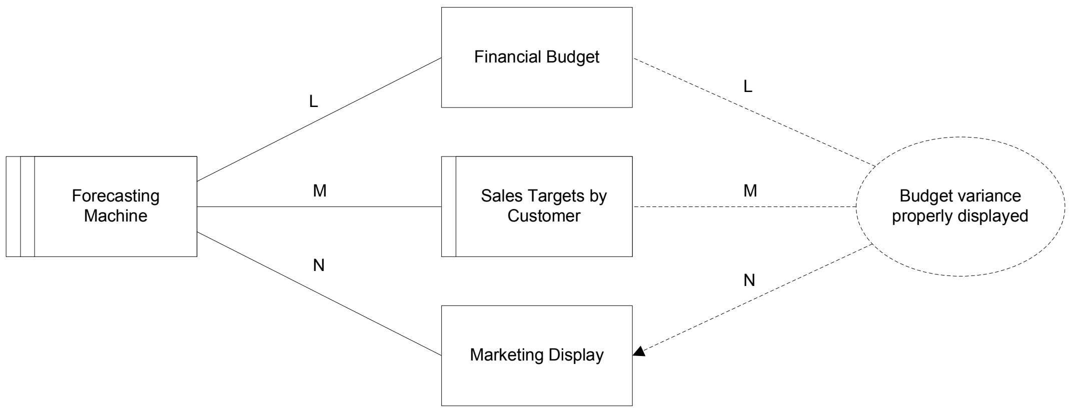
Problem 2 – Diagram 1


Sales Targets by Customer
H
J
Forecasting Machine
Sales Targets updated in accordance with user commands
I
K
Marketing

Interfaces:

Interface	Description
H	FM! {Sales target edit operations}, ST! {Sales target states}
I	MA! {Editing commands}
J	ST! {Sales target effects}
K	MA! {User commands}

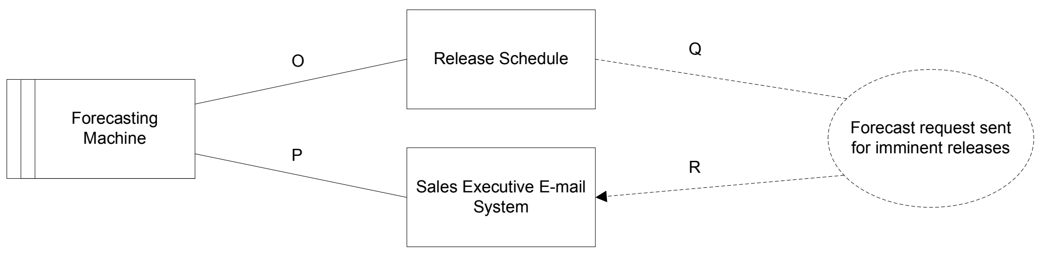
Problem 2 – Diagram 2



Interfaces:

Interface	Description
L	FB! {Budget quantity}
M	ST! {Targeted quantity}
N	FM! {Budget variance}

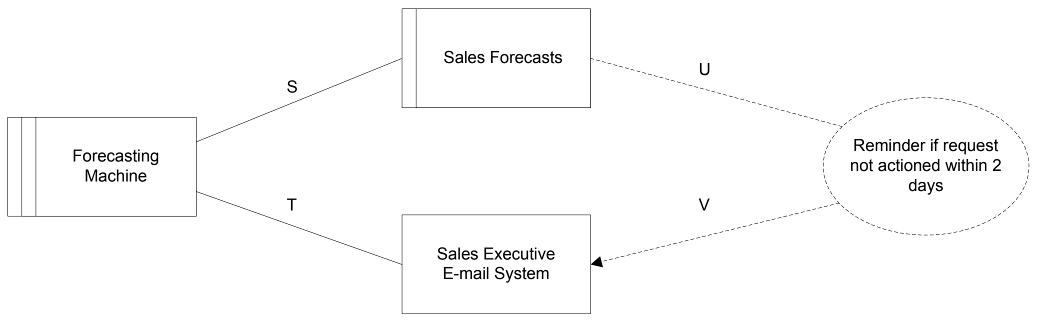
Problem 2 – Diagram 3


O
Release Schedule
Q
Forecasting Machine
Forecast request sent for imminent releases
P
R
Sales Executive E-mail System

Interfaces:

Interface	Description
O	RS! {Scheduled release dates}
P	FM! {Formatted e-mail content}
Q	RS! {Scheduled releases}
R	FM! {E-mail forecast request}

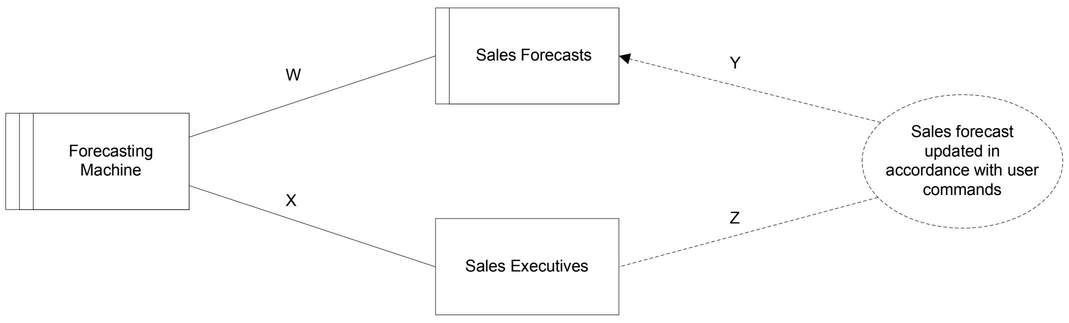
Problem 2 – Diagram 4


Sales Forecasts
S
U
Forecasting Machine
Reminder if request not actioned within 2 days
T
V
Sales Executive E-mail System

Interfaces:

Interface	Description
S	SF! {Uncompleted forecast requests}
T	FM! {Formatted e-mail}
U	SF! {Uncompleted forecasts}
V	FM! {Forecast reminder}

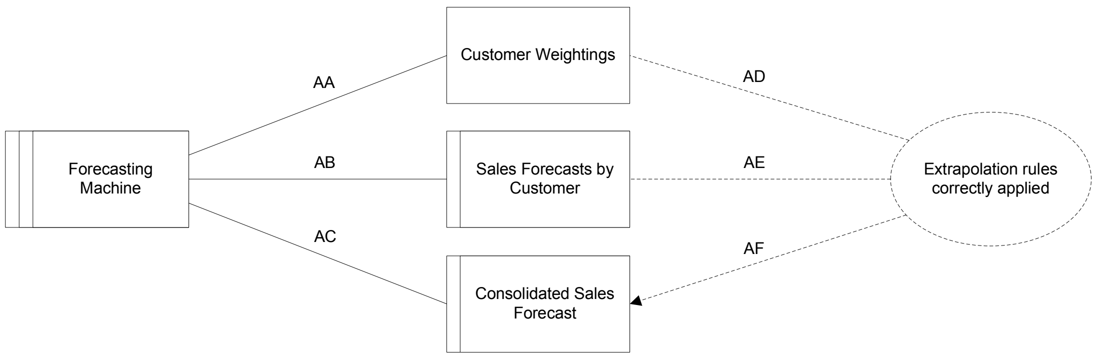
Problem 2 – Diagram 5



Interfaces:

Interface	Description
W	FM! {Sales forecast edit operations}, SF! {Sales forecast states}
X	SE! {Editing commands}
Y	SF! {Sales target effects}
Z	SE! {User commands}

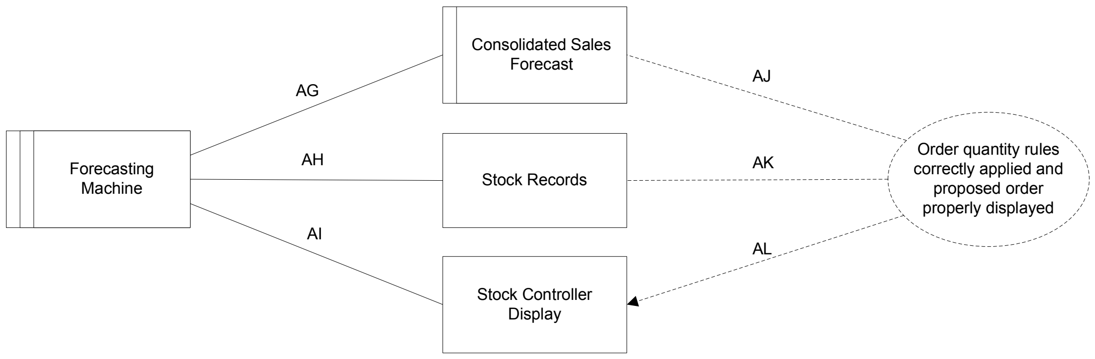
Problem 2 – Diagram 6


Customer Weightings
AA
AD
Forecasting Machine
AB
Sales Forecasts by Customer
AE
Extrapolation rules correctly applied
AC
AF
Consolidated Sales Forecast

Interfaces:

Interface	Description
AA	CW! {customer weightings}
AB	SF! {Sales forecast}
AC	FM! {Calculated total forecast quantity}
AD	CW! {customer weightings}
AE	SF {Entered forecast quantity}
AF	FM! {Consolidated forecast quantity}

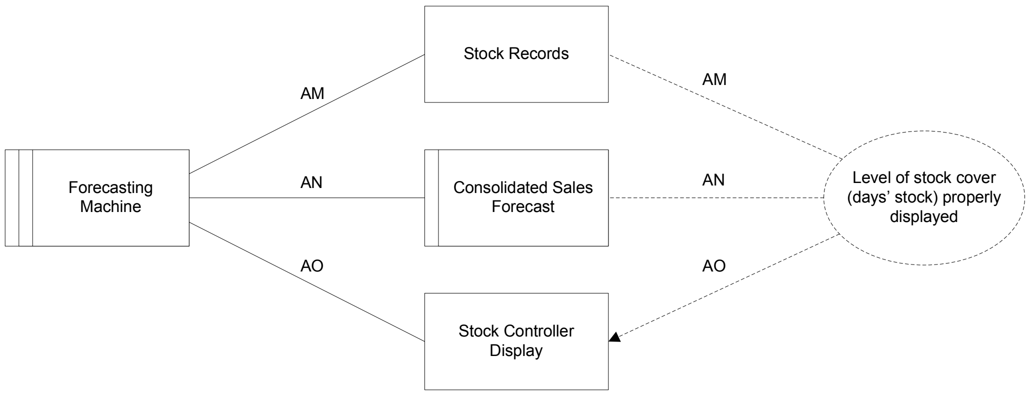
Problem 2 – Diagram 7


Consolidated Sales Forecast
AG
AJ
Forecasting Machine
AH
Stock Records
AK
Order quantity rules correctly applied and proposed order properly displayed
AI
AL
Stock Controller Display

Interfaces:

Interface	Description
AG	CS! {Sales forecast quantity}
AH	SR! {Stock quantity}
AI	FM! {Stock coverage data}
AJ	CS! {Sales forecast}
AK	SR! {Current stock}
AL	SC! {Number of days stock coverage}

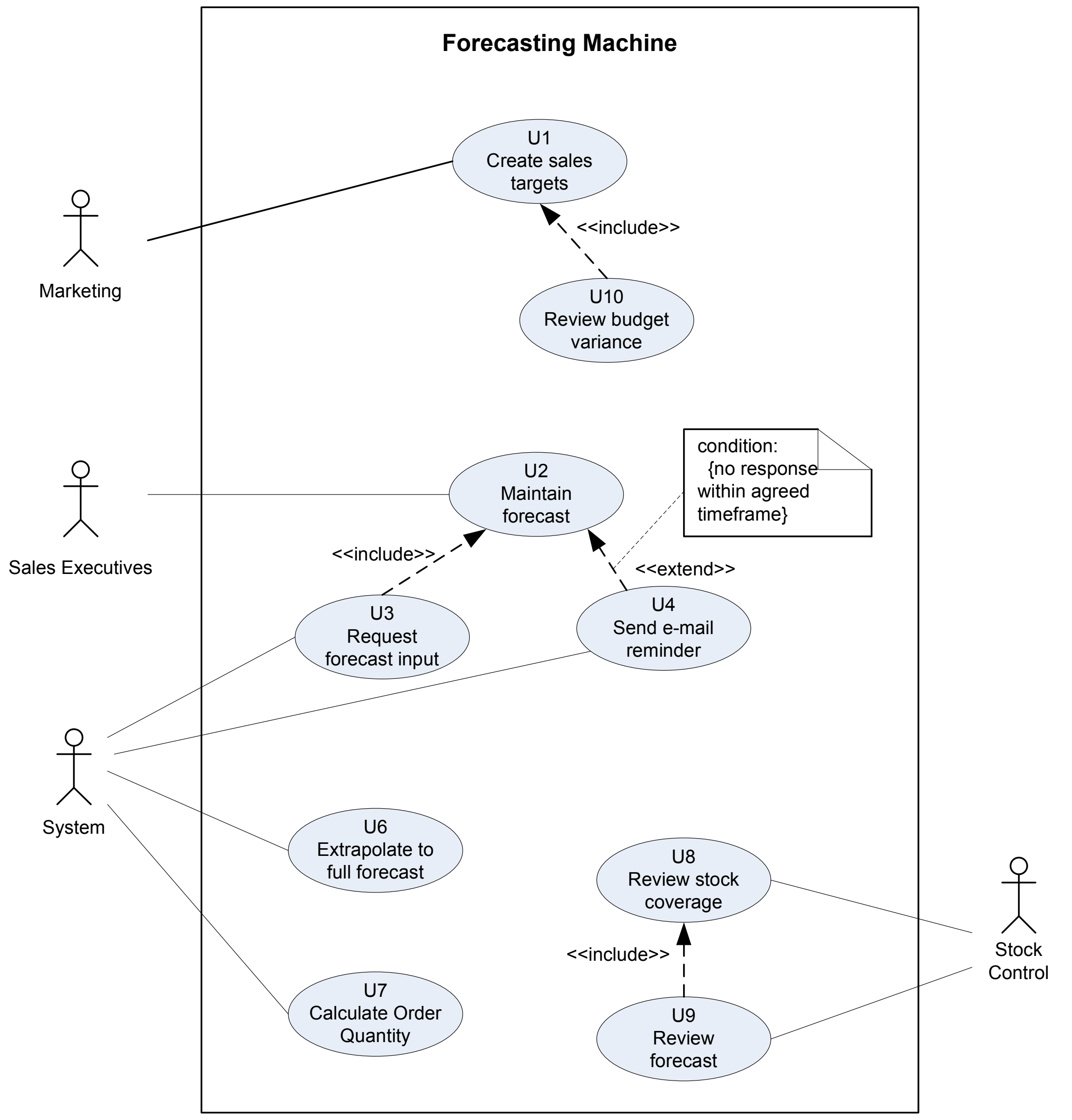
Problem 2 – Diagram 8


Stock Records
AM
AM
Forecasting Machine
AN
Consolidated Sales Forecast
AN
Level of stock cover (days' stock) properly displayed
AO
AO
Stock Controller Display

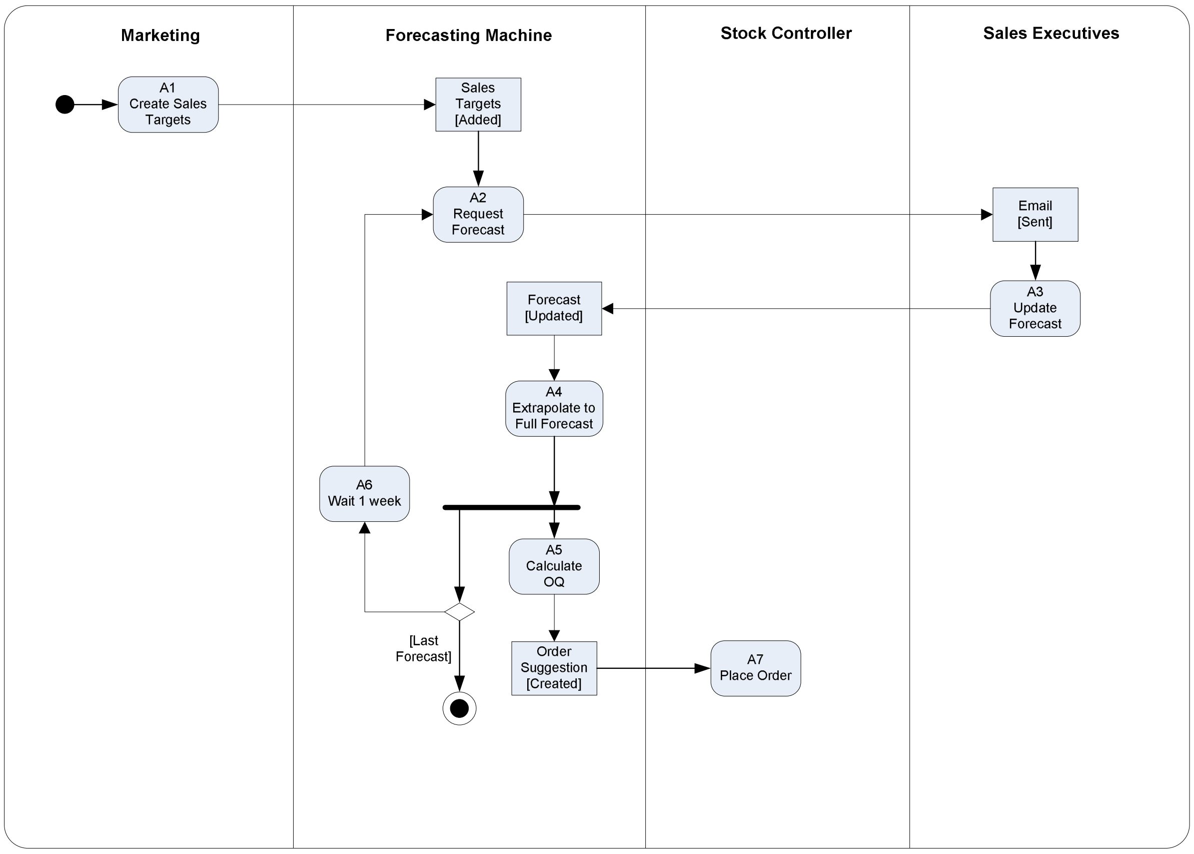
Interfaces:

Interface	Description
AM	FB! {Stock quantity}
AN	ST! {Forecast quantity per week}
AO	FM! {Stock cover in days}

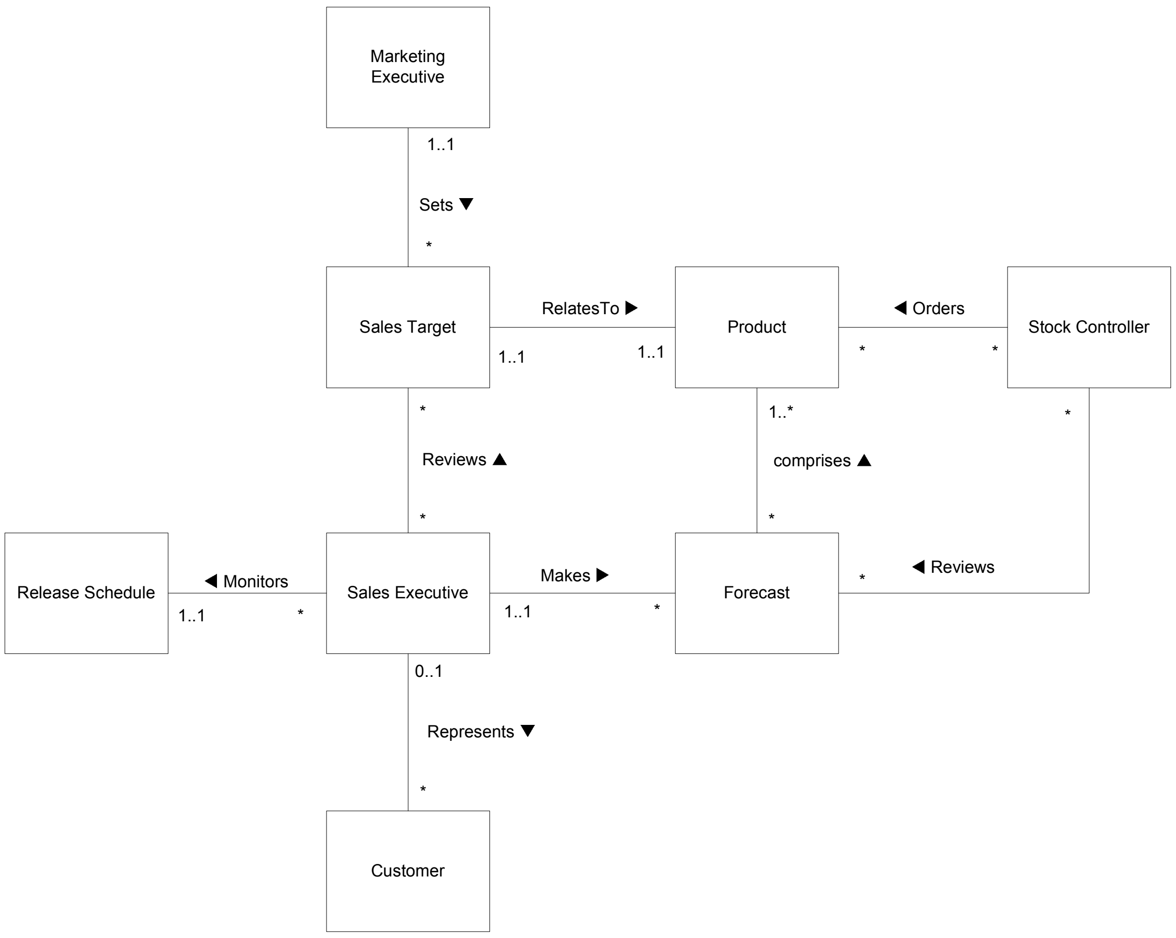
Problem 2 - Use Case View



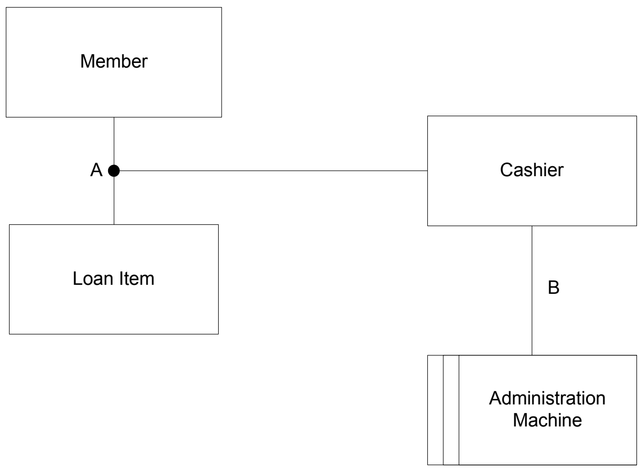
Problem 2 - Activity Diagram



Problem 2 – Class Diagram – Domain concepts



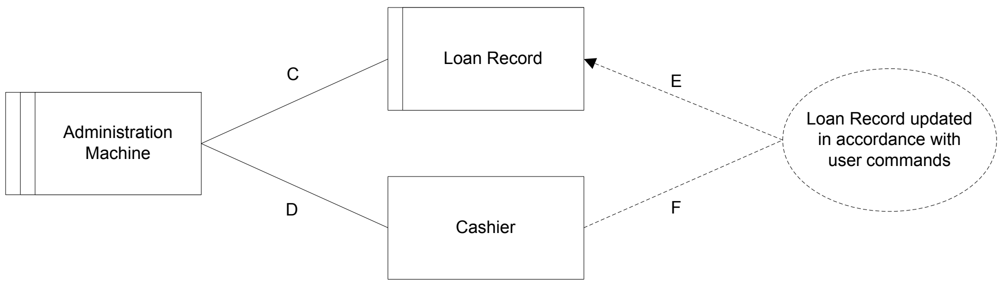
Problem 3 - Context



Interfaces:

Interface	Description
A	Item Loan Request
B	Member and item details, loan request response

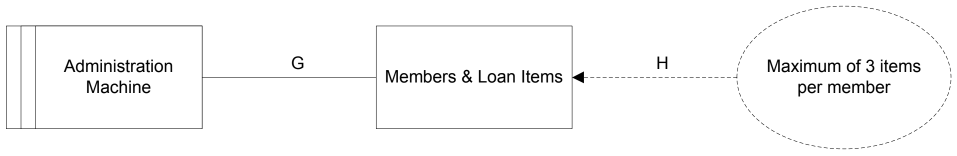
Problem 3 - Diagram 1



Interfaces:

Interface	Description
C	AM! {Loan edit operations}, LR! {Loan record states}
D	LS! {User commands}
E	LR! {Loan record effects}
F	LS! {User commands}

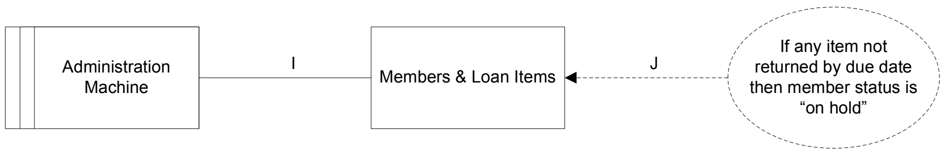
Problem 3 - Diagram 2



Interfaces:

Interface	Description
G	ML! {Items on loan per member}
H	ML! {Allowed items per member}

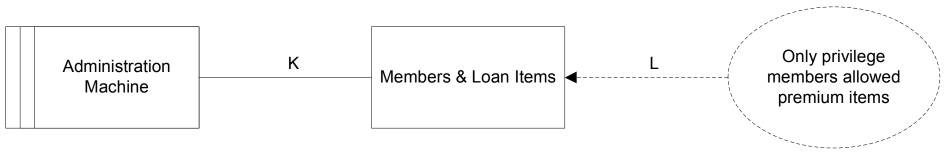
Problem 3 – Diagram 3


Administration Machine
I
Members & Loan Items
J
If any item not returned by due date then member status is “on hold”

Interfaces:

Interface	Description
I	ML! {Open late items}
J	AM! {Member status}

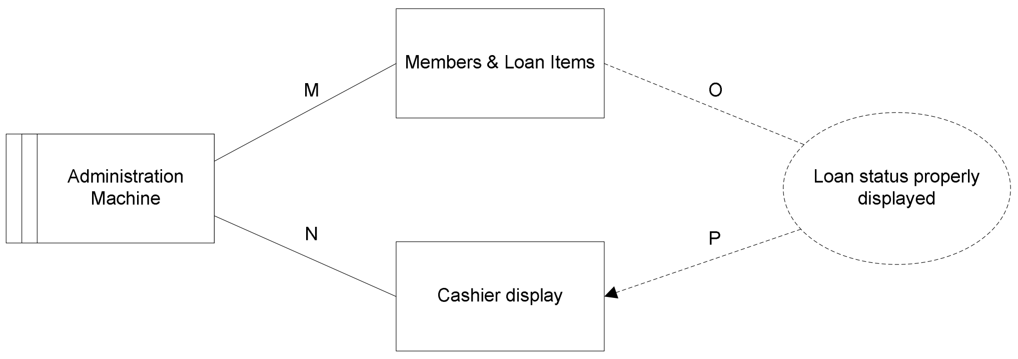
Problem 3 – Diagram 4



Interfaces:

Interface	Description
K	ML! {Loan item class, membership level}
L	AM! {Authorisation}

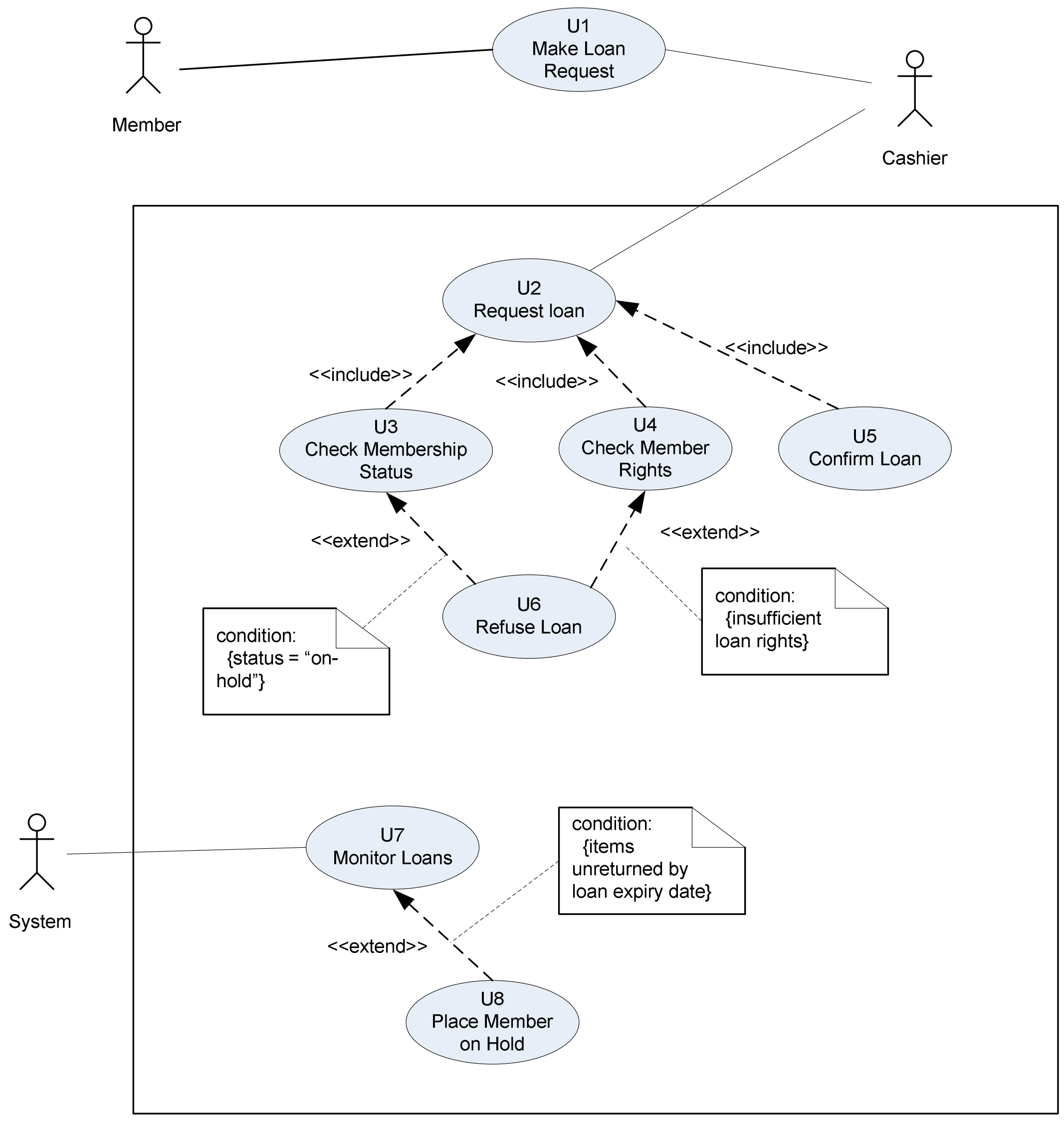
Problem 3 – Diagram 5



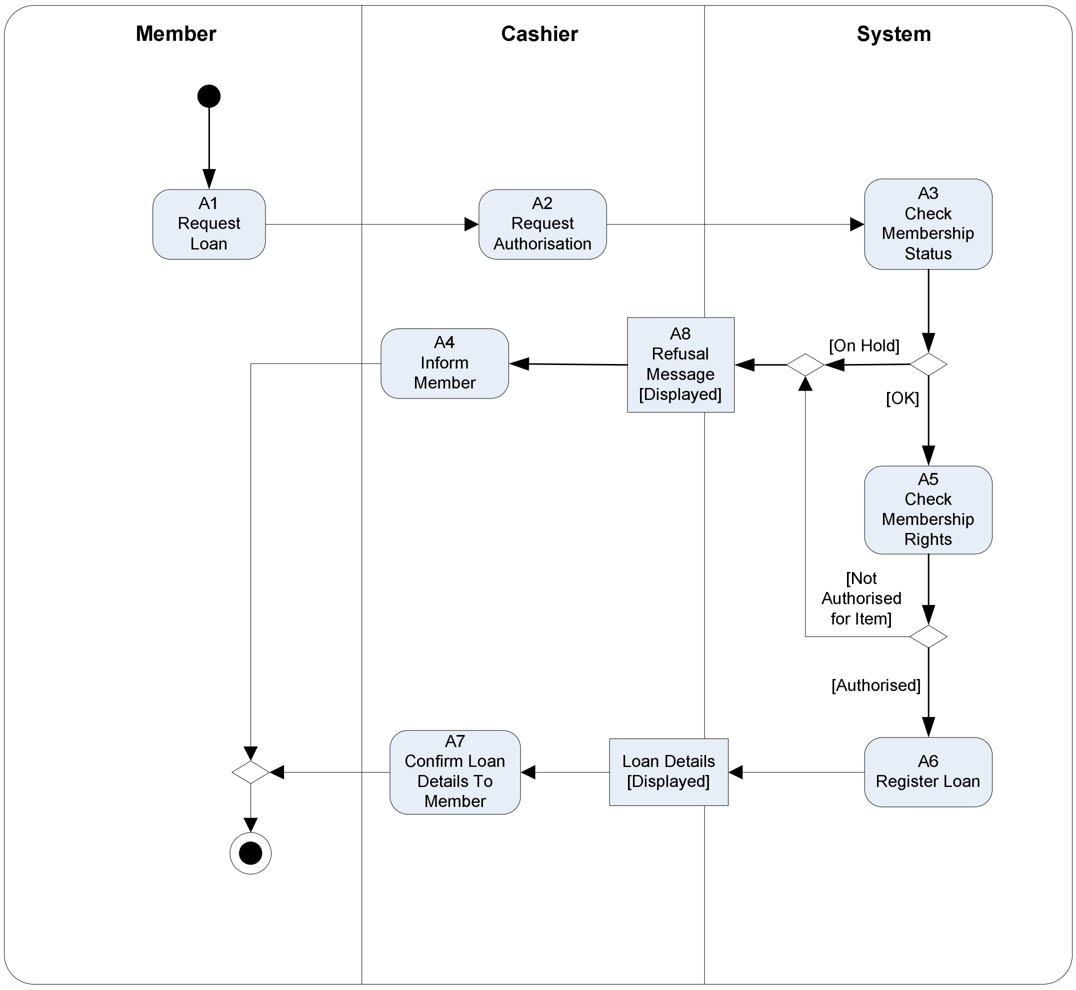
Interfaces:

Interface	Description
M	ML! {Member status, loan item type}
N	AM! {Loan status}
O	ML! {Member status, Loan Item Type}
P	AM! {Loan status}

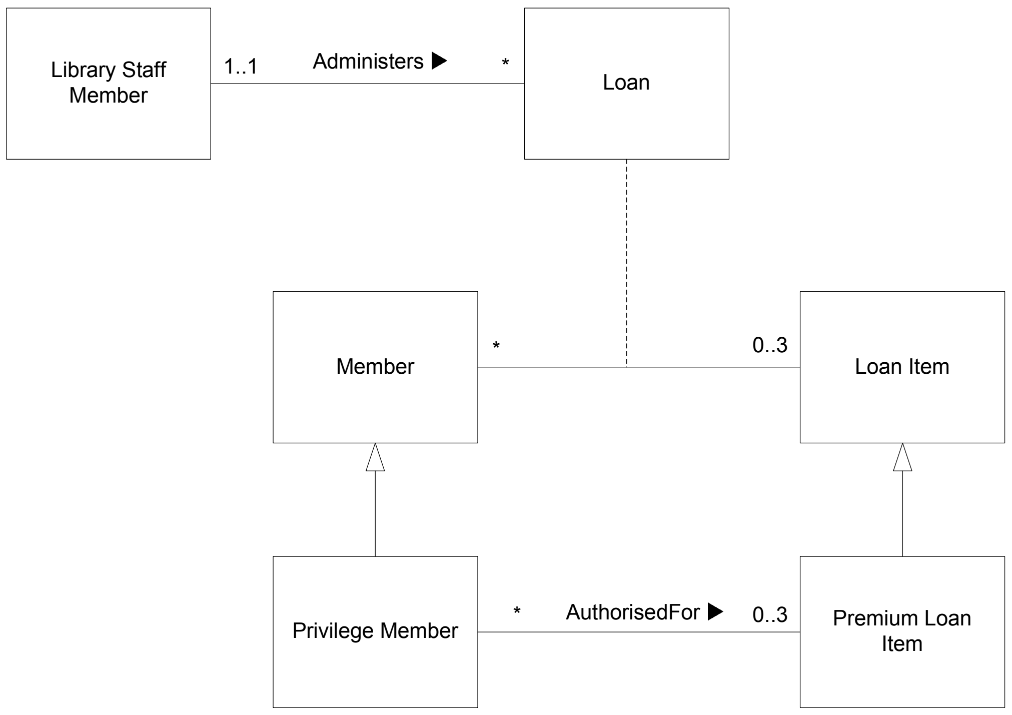
Problem 3 - Use Case View



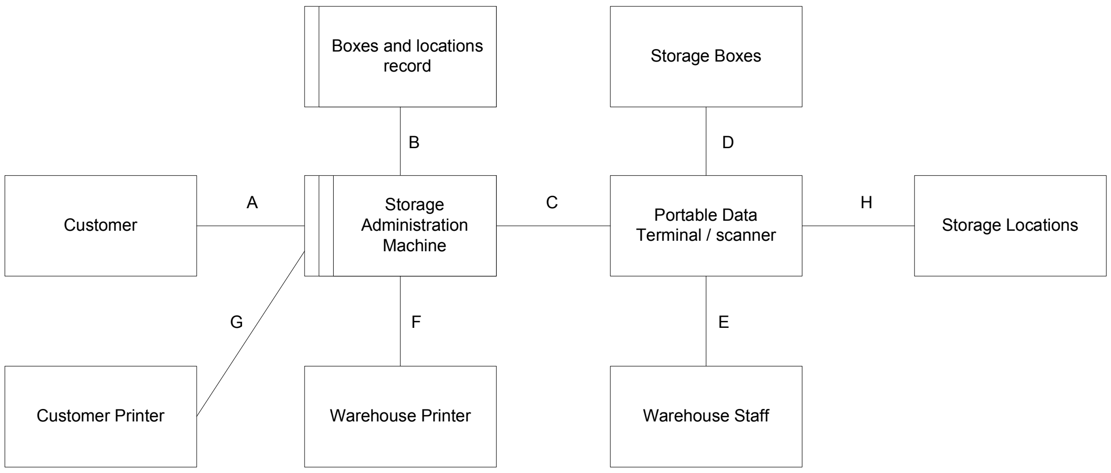
Problem 3 - Activity Diagram


Member
Cashier
System
A1 Request Loan
A2 Request Authorisation
A3 Check Membership Status
[On Hold]
[OK]
A8 Refusal Message [Displayed]
A4 Inform Member
A5 Check Membership Rights
[Not Authorised for Item]
[Authorised]
A6 Register Loan
Loan Details [Displayed]
A7 Confirm Loan Details To Member

Problem 3 – Class Diagram – Domain concepts



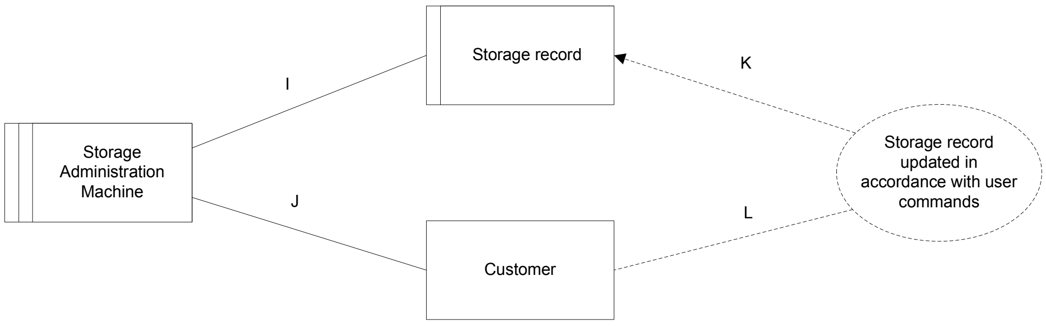
Problem 4 - Context



Interfaces:

Interface	Description
A	Storage request, confirmed box identifiers
B	Box and warehouse location updates, unreserved locations, warehouse layout model
C	Scanned items and locations, suggested location
D	Box barcode label
E	Suggested location
F	Warehouse printed documents
G	Customer printed documents
H	Location barcode label

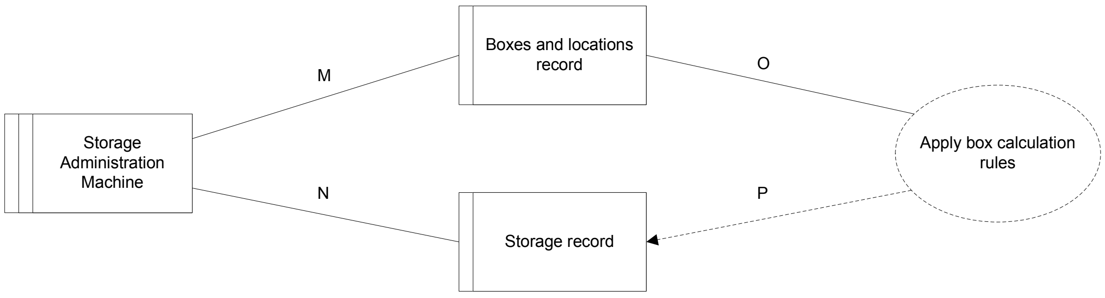
Problem 4 – Diagram 1


Storage record
I
K
Storage Administration Machine
Storage record updated in accordance with user commands
J
L
Customer

Interfaces:

Interface	Description
I	SM! {Storage record edit operations}, SR! {Storage record states}
J	CU! {Editing commands}
K	SR! {Storage record effects}
L	CU! {User commands}

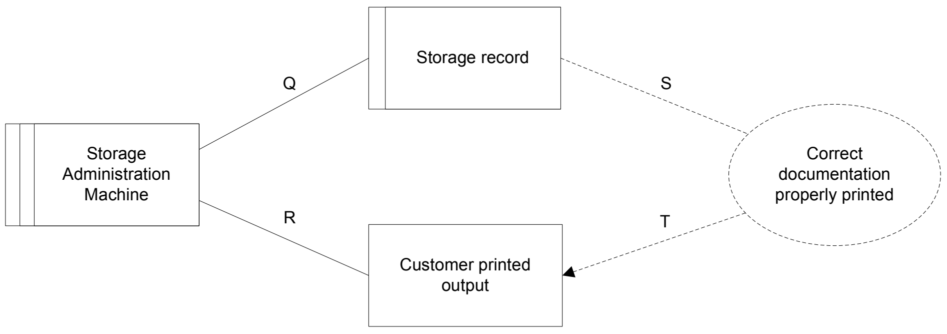
Problem 4 – Diagram 2


Boxes and locations record
M
O
Storage Administration Machine
Apply box calculation rules
N
P
Storage record

Interfaces:

Interface	Description
M	BL! {Warehouse layout model}
N	SM! {Box sizes and unique identifiers}
O	BL! {warehouse layout model}
P	SM! {Box sizes and unique identifiers}

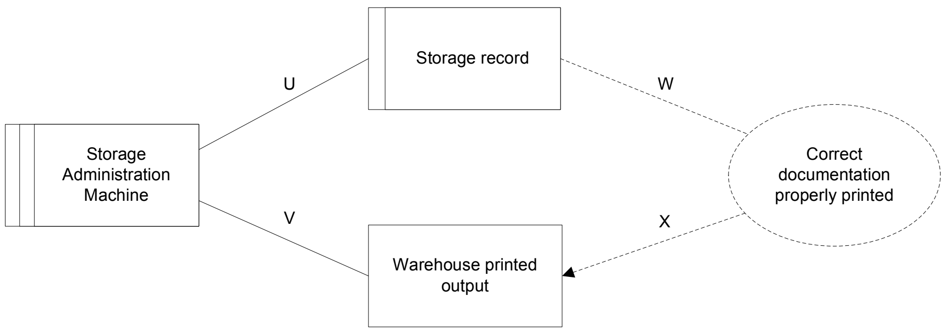
Problem 4 – Diagram 3


Storage record
Q
S
Storage Administration Machine
Correct documentation properly printed
R
T
Customer printed output

Interfaces:

Interface	Description
Q	SR! {Box IDs and quantities},
R	SM! {Label and shipping print stream}
S	SR! {List of boxes and quantities}
T	SM! {Labels and shipping printed output}

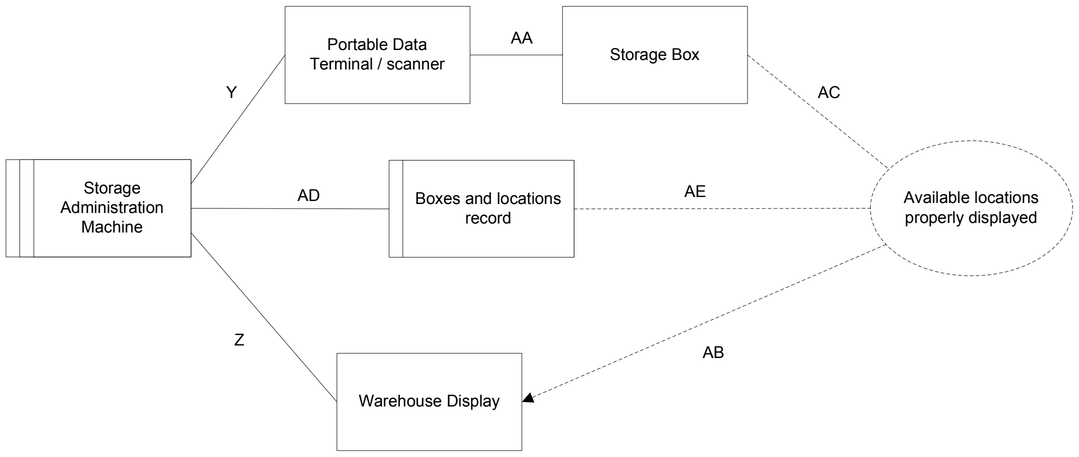
Problem 4 – Diagram 4


Storage record
U
W
Storage Administration Machine
Correct documentation properly printed
V
X
Warehouse printed output

Interfaces:

Interface	Description
U	SR! {Box IDs and quantities},
V	SM! {Packing list & delivery note print stream}
W	SR! {List of boxes and quantities}
X	SM! {Packing list and delivery note printed output}

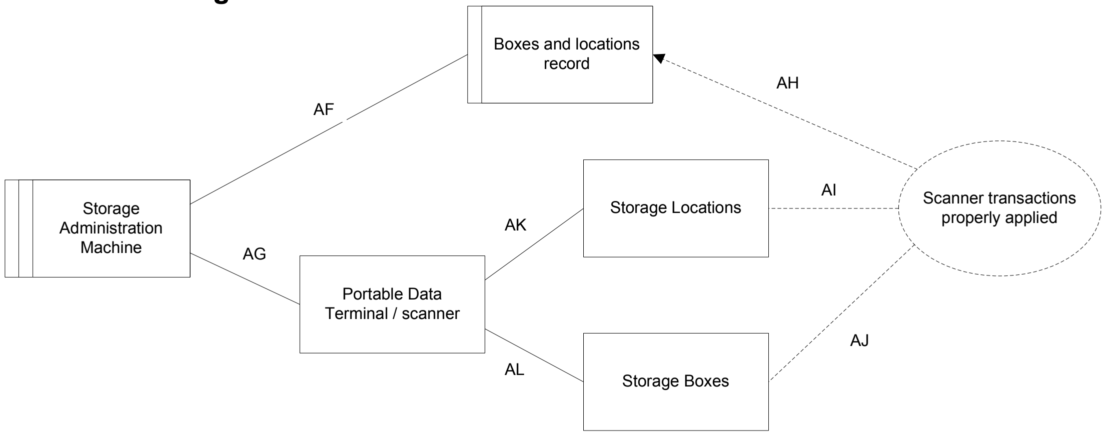
Problem 4 - Diagram 5


Portable Data Terminal / scanner
AA
Storage Box
Y
AC
Storage Administration Machine
AD
Boxes and locations record
AE
Available locations properly displayed
Z
AB
Warehouse Display

Interfaces:

Interface	Description
Y	PD! {Scan data stream}
Z	SM! {Storage location display stream}
AA	SB! {Box barcode}
AB	WD! {Proposed storage location}
AC	SB! {Box size}
AD	BL! {warehouse model data-stream}
AE	BL! {Free locations and sizes}

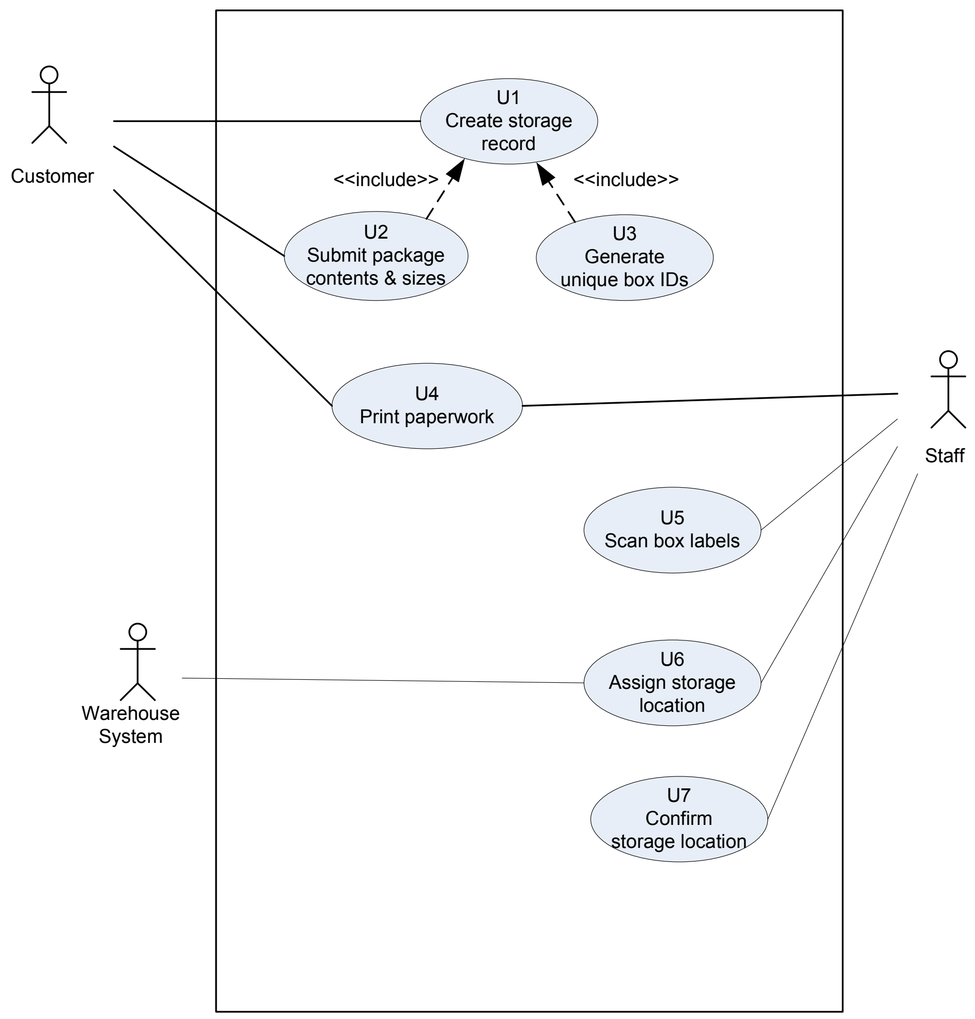
Problem 4 – Diagram 6


Boxes and locations record
AH
AF
Storage Locations
AI
Scanner transactions properly applied
Storage Administration Machine
AK
AG
Portable Data Terminal / scanner
AJ
AL
Storage Boxes

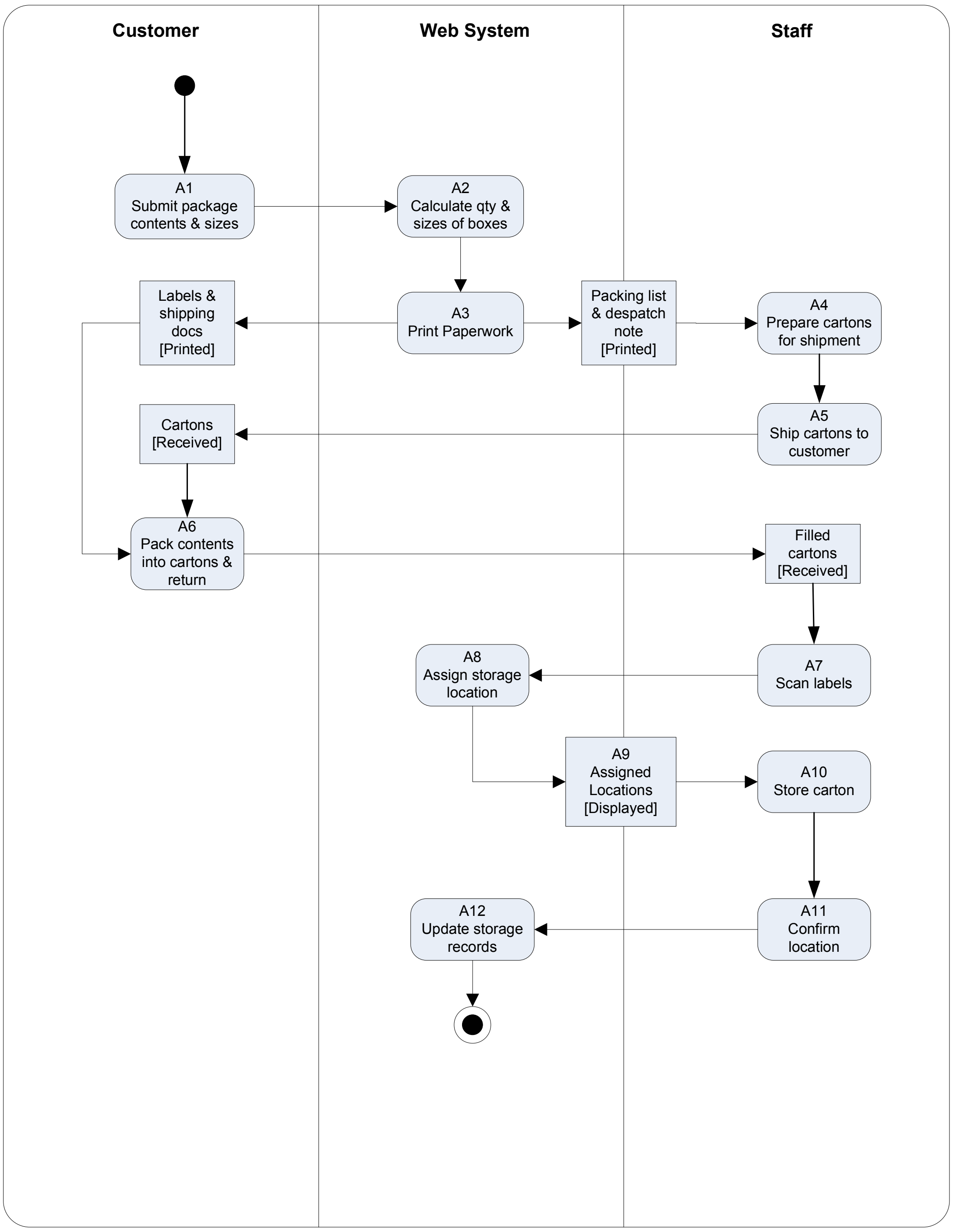
Interfaces:

Interface	Description
AF	SM! {Edit operations}, BL! {Boxes and locations record states}
AG	PD! {scan data stream}
AH	BL! {Boxes and locations record effects}
AI	SL! {Location identifier}
AJ	SB! {Box identifier}
AK	SL! {Location barcode}
AL	SB! {Box barcode}

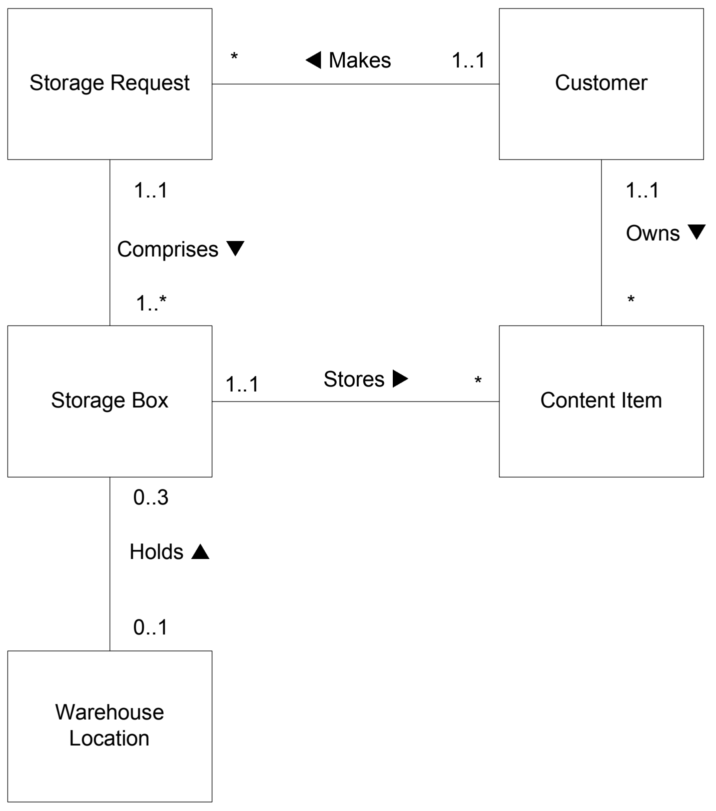
Problem 4 - Use Case View


Customer
Staff
Warehouse System
U1 Create storage record
U2 Submit package contents & sizes
U3 Generate unique box IDs
<<include>>
<<include>>
U4 Print paperwork
U5 Scan box labels
U6 Assign storage location
U7 Confirm storage location

Problem 4 - Activity Diagram


Customer
Web System
Staff
A1 Submit package contents & sizes
A2 Calculate qty & sizes of boxes
A3 Print Paperwork
Labels & shipping docs [Printed]
Packing list & despatch note [Printed]
A4 Prepare cartons for shipment
A5 Ship cartons to customer
Cartons [Received]
A6 Pack contents into cartons & return
Filled cartons [Received]
A7 Scan labels
A8 Assign storage location
A9 Assigned Locations [Displayed]
A10 Store carton
A11 Confirm location
A12 Update storage records

Problem 4 – Class Diagram – Domain concepts



Appendix E – Comprehension Scores and Perceived Difficulty Ratings

			Participant									
Problem	**Requirement**	**Baseline**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	R1	**1**	0.5	1	1	1	0	1	0.5	0	0	0
1	R2	**1**	0.5	1	0	0	0	1	1	0.5	1	1
1	R3	**1**	0.5	1	0	1	0.5	1	1	0.5	1	1
1	R4	**1**	1	0	0	1	0	0.5	1	1	0	0.5
1	R5	**1**	0	1	0	0	0	1	1	0.5	1	1
1	R6	**1**	1	0	0	0	0	1	1	1	1	1
	Score	**6**	**3.5**	**4**	**1**	**3**	**0.5**	**5.5**	**5.5**	**3.5**	**4**	**4.5**
	Rating	**5**	**2**	**3**	**2**	**4**	**3**	**4**	**4**	**4**	**2**	**4**
	Notation		**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**
2	R1	**1**	0.5	1	1	1	0.5	1	0.5	0.5	1	0
2	R2	**1**	1	0	1	0.5	0	1	0	0	1	1
2	R3	**1**	0	0	0	0	0	1	0	0	0.5	1
2	R4	**1**	1	0	1	0.5	0.5	1	0	0	1	1
2	R5	**1**	1	0	0	0	0	1	0	0	1	0.5
2	R6	**1**	1	0	0	0	0	1	0	0.5	1	1
2	R7	**1**	0	1	0	0	0	1	0	0.5	1	0
	Score	**7**	**4.5**	**2**	**3**	**2**	**1**	**7**	**0.5**	**1.5**	**6.5**	**4.5**
	Rating	**5**	**4**	**4**	**5**	**2**	**2**	**3**	**2**	**3**	**4**	**2**
	Notation		**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**
3	R1	**1**	0.5	0.5	0	1	0	1	0.5	0.5	1	1
3	R2	**1**	1	0.5	1	0.5	0	0	0.5	0.5	1	0
3	R3	**1**	1	1	1	0	0	1	1	1	1	1
3	R4	**1**	1	0.5	1	1	0	1	1	0	1	0.5
3	R5	**1**	0.5	0.5	0	0	0	1	0	1	1	1
	Score	**5**	**4**	**3**	**3**	**2.5**	**0**	**4**	**3**	**3**	**5**	**3.5**
	Rating	**5**	**4**	**3**	**3**	**4**	**1**	**4**	**4**	**4**	**2**	**4**
	Notation		**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**
4	R1	1	1	0.5	1	0.5	0.5	1	1	1	1	0.5
4	R2	1	1	0	0.5	0	0.5	1	0.5	0.5	1	0
4	R3	1	0	0	1	0.5	0	1	1	0.5	0.5	0
4	R4	1	0	0	1	0.5	0	1	1	0.5	1	0
4	R5	1	0	0.5	1	0.5	0	1	0	0	1	0
4	R6	1	0.5	0	0	0	0	0.5	0	1	1	0
	Score	**6**	**2.5**	**1**	**4.5**	**2**	**1**	**5.5**	**3.5**	**3.5**	**5.5**	**0.5**
	Rating	**5**	**5**	**2**	**5**	**2**	**3**	**4**	**3**	**3**	**5**	**1**
	Notation		**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**	**UML**	**PF**